アプリケーションの特性に応じた
計算構造による
リアルタイム動画像処理の研究

2020年12月

長崎大学大学院工学研究科

眞邉 泰斗

目 次

第 1 章

緒論

近年，ニューラルネットワーク (Neural Network) に代表される機械学習手法の発展と，アクセラレータとして広く用いられている GPU の演算性能の向上に伴って，大量のデータに内在する特徴を活用した様々なサービスが実現できるようになった．とりわけ画像に対しては，2 次元畳み込みニューラルネットワーク (Convolutional Neural Network, CNN) の活用により、写真から人物やペットの顔を高精度に検出，分類する，特定の被写体を取り除き欠落部分を自動的に補間する，パラメータを与えることにより自然な画像を人工的に生成する [1, 2] 等の高度な機能を持つサービスがすでに実用に供されており，今なお目覚ましい勢いで発展を続けている．

その一方で，肥大化を続ける畳み込みニューラルネットワークでは，膨大な積和演算による演算負荷とメモリ帯域の圧迫，さらにはそれに伴う大きな消費電力が問題となっており，動画像をリアルタイムに扱うような場合には、いまだ大きな困難がつきまとう。特に，演算性能や電力面に制約のある組み込み機器での運用では，問題はより深刻となる．そこで，演算量やパラメータ数を削減するための研究が盛んに行われており，例えば，畳み込み演算をより小さな演算に分解する [3, 4, 5, 6]，量子化や枝刈り (プルーニング, Pruning) により演算そのものを簡略化する [7, 8, 9, 10] 等の手法が挙げられるが，CPU や GPU を用いたソフトウェア処理では，演算器の柔軟性が十分でない，メモリアクセスに律速される等の理由により，理論上期待されるほどの高速化が得られないことが多い．また，畳み込みニューラルネットワークを用いたシステムの多くは，学習のために非常に大規模なデータセットを必要としており，そのようなデータセットを用意できない分野における利用には大きな制限がある．加えて，畳み込みニューラルネットワークのネットワークモデルは日々進化を続けているが，万能と呼べるようなモデルはなく，用途に応じて評価を繰り返しながら適切なモデルを選択，あるいは構築する必要があるため，さらなる知見の蓄積が望まれている．

そこで本研究では，動画像を用いる二つのアプリケーションを対象に，それぞれの特性に応じたネットワークの構成および演算手法を提案し，その性能について評価を行う．その一つは，低解像度画像から高解像度画像を再構成する超解像 (Super Resolution) である．畳み込みニューラルネットワークを用いた超解像にはいくつかの手法が提案されている [11, 12, 13, 14] が，GPU を用いた実装が主流であり，テレビジョンなどの組み込み機器での活用には，消費電力やレイテンシの点で課題がある．そこで本研究では，Field-Programmable Gate Array (FPGA) を用いた完全パイプライン実装により，これらの問題に対処する．またもう一つのアプリケーションは，腹腔鏡手術 (Laparoscopic Surgery)

の支援を目的とする，手術画像を対象としたセグメンテーションである．このアプリケーションでは，データセットのアノテーションに専門的な知識と経験を持つ医師の協力が不可欠であることから，大規模なデータセットの用意が難しく，ネットワークが過学習に陥りやすいことが大きな問題となる．そこで本研究では，過学習のリスクを低減し，小さなデータセットでも品質を向上させやすいネットワークモデルを提案する．

また，ニューラルネットワークを用いないアプリケーションとして，マイクロサージェリーにおける執刀医の振戦抑制を実現するため，動画像から振動成分をリアルタイムに検出，抽出するシステムの設計と実装も行う．ニューラルネットワークを用いた高度な推論システムは，広義には人工知能 (Artificial Intelligence, AI) とも呼ばれ，多くの注目を集めていることから，様々な分野で適用可能性が模索されている．しかしながら，すでに述べた演算量の大きさやデータセットの構築にまつわる問題に加えて，その動作メカニズムや性能に理論的な裏付けを与えることが難しく，また，多様な環境における頑健性の懸念もある．実際，ニューラルネットワークを欺く攻撃手法も発表されている [15]．以上のことから，人手で設計された特徴量の使用も依然として重要であると考えられ，このアプリケーションは，その例の一つである．

本稿の構成は以下に示すとおりである．まず第 2 章にて，本研究の基礎を成すいくつかの技術，理論に関する簡単な説明を行う．続く第 3 章では，完全パイプライン構造の畳み込みニューラルネットワークを用いた，リアルタイム動画像超解像システムの設計と FPGA 実装について説明する．また第 4 章では，再帰的構造を導入することで過学習の軽減を図った，畳み込みニューラルネットワークによる手術画像セグメンテーションシステムの設計を扱う．さらに第 5 章で，Lucas-Kanade オプティカルフローと Band-Limited Multiple Fourier Linear Combiner を組み合わせた画像ベースのリアルタイム振戦検出の設計と FPGA 実装について述べ，最後に第 6 章にて本稿全体の結論を述べる．

第2章

理論

本章では，本研究の基礎となる各種の技術や理論について概説する．

2.1 FPGA

Field-Programmable Gate Array (FPGA) は，構成を電気的に変更して任意の論理回路を実現できる集積回路であり，プログラマブル・ロジック・デバイス (Programmable Logic Device, PLD) の一種である [16]．1985 年に米 Xilinx 社が初めて製品化した．

典型的な FPGA は，プログラマブルな論理ブロックを多数備えている．一般的に，論理ブロックは複数の論理セルから成り，論理セルは，複数入力のルックアップテーブル (Look-Up Table, LUT) や D 型フリップフロップ等で構成されている．LUT はスタティックメモリ (SRAM) を用いて実装したものが主流であり，これを書き換えることにより，論理積，論理和，その他様々な論理を表現できる．LUT の入力数 (アドレスのビット幅) は回路規模や動作周波数のバランスに関わり，ベンダや製品によって異なるが，Xilinx 社の現行製品では主に 6 入力 LUT が用いられている．単独の LUT で表現できない複雑な論理は，複数の LUT を組み合わせて実現される．また，LUT を小容量のメモリ (分散メモリ) として利用することもできる．

FPGA には，論理ブロックの他にも，外部との信号入出力を行う I/O ブロックや，論理ブロックおよび I/O ブロックの間に任意の配線経路を形成できる配線要素が備わっている．また多くの場合，テスト回路や組み込みプロセッサ，論理ブロックで構成するとコストが大きいメモリや乗算器，DSP ブロックといった固定機能回路も内蔵されている．

FPGA の構成設計は，一般的に，Verilog-HDL，SystemVerilog や VHDL といったハードウェア記述言語 (Hardware Description Language, HDL) を用いたレジスタ転送レベル (Register Transfer Level, RTL) で行う．また，C 言語等による抽象度の高い記述から RTL への変換を行う高位合成 (High-Level Synthesis, HLS) も広く利用されるようになりつつあるが，効率の良いハードウェアを実現するためには，ディレクティブによる並列処理の明示等が必要である場合も多く，ハードウェアの構造を意識せずにアルゴリズムを記述できる水準には至っていないため，さらなる改善が望まれるところである．

用途に合わせて設計される集積回路である ASIC (Application Specific Integrated Circuit) と比べ，FPGA は純粋な集積密度や電力効率，動作速度では劣る一方，開発及び製造期間を短縮でき，設計の変更も容易である．また，近年の先端半導体プロセスは高コストであり，少量生産品においては採用が難しいが，汎用品である FPGA であればその恩

恵を受けられるため，実製品における性能差は縮まると考えられる．こういった背景から，近年 FPGA は，家電製品等への搭載に留まらず，Microsoft 社の Catapult [17] に見られるように，高い電力効率を活かしてデータセンター等のアクセラレーションにも用いられるようになり，応用範囲が広がっている．

2.2 ニューラルネットワーク

ニューラルネットワーク (Neural Network) は，生物の脳に見られる，多数のニューロン (神経細胞) が他のニューロンと結びついて情報処理を行う仕組みに着想を得て構築された数学モデルの一種である．

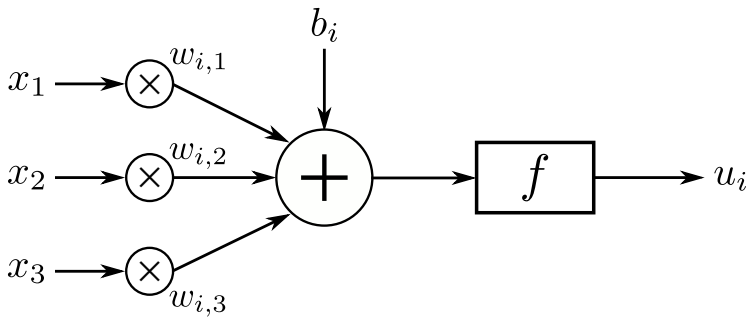


図 2.1: ニューラルネットワークにおけるノードの模式図

ニューラルネットワークには，生体におけるニューロンに相当するノードが複数存在し，相互に結びついている．ノードの模式図を図 2.1 に示す．ノード i の出力 u_i は，通常，n (≥ 1) 個の入力 x_j ($1 \leq j \leq n$) に基づいて，以下の式により決定される．

$$u_i = f\left(\sum_{j=1}^{n} x_j w_{i,j} + b_i\right) \tag{2.1}$$

ただし，$w_{i,j}$ は入力 x_j に対する重み，b_i はバイアスと呼ばれ，いずれもノードごとに固有の値をとる．また，f は活性化関数 (Activation Function) と呼ばれる関数であり，以下のようなものが広く用いられている．

- 標準シグモイド関数 $f(x) = \frac{1}{1+e^{-x}}$
- 双曲線正接関数 $f(x) = \tanh(x)$
- ReLU [18, 19, 20] $f(x) = \max(0,\ x)$
- Leaky ReLU [21] $f(x) = \max(ax,\ x)\ \ (0 < a < 1)$

活性化関数は，ニューロンの発火現象を模擬するものである．活性化関数として線形関数を用いた多層ネットワークには，等価な単層ネットワーク (隠れ層を持たないネットワーク) が存在することが知られており，通常，活性化関数には非線形関数を用いる．

ニューラルネットワークの構造には種々のものがあるが，ここでは代表的な順伝播型ニューラルネットワーク (Feedforward Neural Network) についてのみ述べる．このモデルでは，ノード間の結合にループが存在せず，信号は入力層から隠れ層，そして出力層へと一方向に流れる．順伝播型ニューラルネットワークの一種である多層パーセプトロン (Multilayer Perceptron) の例を図 2.2 に示す．隠れ層が 1 層以上存在し，非線形の活性化関数を用いた多層パーセプトロンは，ノード数が十分であれば，任意の連続関数を近似可能であることが知られている．

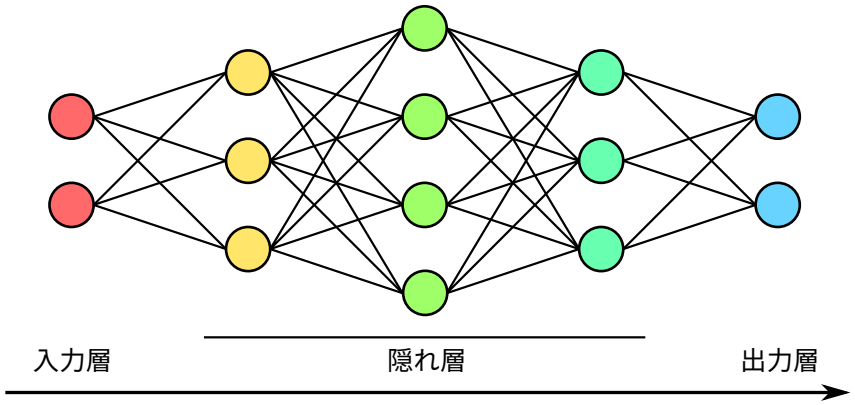


図 2.2: 多層パーセプトロンの例

2.3　ニューラルネットワークの学習

ニューラルネットワークを問題解決に利用するためには，各ノードの持つパラメータ (重みおよびバイアス) を，期待される出力が得られるように調整する必要がある．この調整の過程を一般に学習 (Learning) あるいは訓練 (Training) と呼び，学習済みのパラメータを用いて問題解決を行うことを推論 (Inference) と呼ぶ．学習は，教師あり学習 (Supervised Learning) と教師なし学習 (Unsupervised Learning) とに大別されるが，ここでは教師あり学習についてのみ述べる．教師あり学習においては，事前に用意した訓練データ，および各訓練データに対応する教師データを用い，ネットワークに訓練データを入力した際の出力が教師データに近づくように，パラメータを調整していく．その手法として繁用されているものが，誤差逆伝播法 (Backpropagation) [22] である．

まず，ネットワークに訓練データを与え，得られた出力と教師データとの誤差を求める．回帰モデルでは一般的に，誤差関数として平均二乗誤差

$$E = \frac{1}{n}\sum_{i=1}^{n}(t_i - o_i)^2$$

等を用いる．ただし，n は出力ノード数であり，o_i 及び t_i は，それぞれ訓練データに対する出力ノード i の出力及び教師データである．また，多クラス分類モデルでは，ソフト

マックス交差エントロピー (Softmax Cross-Entropy)

$$E = -\sum_{i=1}^{n} t_i \log p_i$$

等を用いる．ただし，p_i は，出力 o_i を分類結果がクラス i である確率とみなせるように，ソフトマックス関数を用いて正規化したものであり，以下で与えられる．

$$p_i = \frac{\exp(o_i)}{\sum_{k=1}^{n} \exp(o_k)}$$

また，出力ノード数 n は分類クラス数に等しく，教師データ t_i は，正解クラスでは 1，それ以外のクラスでは 0 をとる one-hot のラベルである．ソフトマックス交差エントロピーは，確率分布 t_i と p_i のカルバック・ライブラー情報量 (Kullback-Leibler Divergence)

$$D = \sum_{i=1}^{n} t_i \log \frac{t_i}{p_i} = \sum_{i=1}^{n} (t_i \log t_i - t_i \log p_i)$$

から定数である $\sum t_i \log t_i$ を取り除いたものであり，これが小さければ小さいほど，p_i は真の確率分布 t_i に近いことを意味している．

続いて，得られた誤差に基づき，誤差がより小さくなるように各出力ノードの重み及びバイアスを更新する．また，現在の重みに基づいて誤差を入力層側へと逆伝播させていき，隠れ層及び入力層のノードについてもパラメータを更新していく．

訓練データ全体に対する誤差の平均に基づいてパラメータを更新する手法をバッチ学習，逆に，訓練データを 1 つずつ与えて逐次パラメータを更新する手法をオンライン学習と呼ぶ．バッチ学習は一般に精度が高いが，処理に時間を要し，メモリ使用量が大きいことからデータセットが巨大である場合には適用が難しい．一方オンライン学習は，大規模なデータセットでも実行できるが，パラメータの更新量を調整する学習率を適切に制御しなければ，学習が収束しないことがある．両者の中間に位置するものとして，複数の訓練データをまとめて与えパラメータを更新するミニバッチ学習も広く用いられている．まとめて与える訓練データの数をミニバッチサイズと呼ぶ．

多層ネットワークの学習を，特に深層学習 (Deep Learning) と呼ぶ．深層学習においては，現在のノードの重みや，選択した活性化関数の種類により，誤差が逆伝播する過程において値が非常に小さくなる (勾配消失)，あるいは発散してしまう (勾配爆発) ことがあり，問題となる．正領域において勾配が常に 1 である ReLU のような活性化関数では，勾配消失及び爆発が起こりづらく，より深いネットワークでも学習が行いやすいため，高い性能を実現できると考えられている [20]．また，ネットワークの各層において，入力値を出力側へとバイパスし，出力値と足し合わせるためのショートカット接続を設ける手法も提案されている [23]．この場合，ネットワークの出力は「本来求められる出力と入力の差分」となる．この手法の導入により，学習時にショートカット接続を介して誤差が直接逆伝播できるようになるため，勾配消失が軽減され，深いネットワークの学習が容易になる．

深層学習では，ネットワークの表現力が高まるため，ネットワークが訓練データに過剰に適応してしまい，推論時に適切に機能しなくなる過学習 (Overfitting) が起こりやすくなることも問題である．これを回避するため，ネットワークの表現力に制限を加える正則

化 (Regularization) 手法が多く提案されている．例として，L2 ノルム等を用いて出力の大きさにペナルティを与えるシンプルな手法のほか，学習時に一部のノードをランダムに無効化する Dropout [24]，各層の出力を正規化する Batch Normalization [25] 等が挙げられる．

2.4　畳み込みニューラルネットワーク

畳み込みニューラルネットワーク (Convolutional Neural Network，CNN) [26] とは，各ノードが全入力に対応する重みを持つ代わりに，フィルタあるいはカーネルと呼ばれる，サイズが入力と同じかより小さなベクトルを備え，入力とフィルタの畳み込み演算に基づいて出力を決定する，ニューラルネットワークの一形態である．

CNN の適用対象は画像に限られるものではないが，以下では説明を簡単にするため，2 次元画像を扱う場合のみを考えることとする．$n \times n$ サイズの入力画像 $X(i,\ j)$ と，$m \times m$ サイズのフィルタ $F(i,\ j)$ との畳み込み演算は以下のように定義できる．

$$(X * F)(i,\ j) = \sum_{k=0}^{m-1} \sum_{l=0}^{m-1} X(i+k,\ j+l) F(k,\ l)$$

ただし，演算後の画像は $(n-(m-1)) \times (n-(m-1))$ サイズであり，i，j の範囲は $0 \leq i \leq n-m$，$0 \leq j \leq n-m$ となる．このように，畳み込み演算により画像サイズが小さくなっていくため，必要に応じて，入力画像に対し画像の周辺部を拡張するパディングを適用する．パディングには，単に 0 で埋めるゼロパディングのほか，最も近い画素値の拡張等，アプリケーションに応じて複数の手法が用いられている．

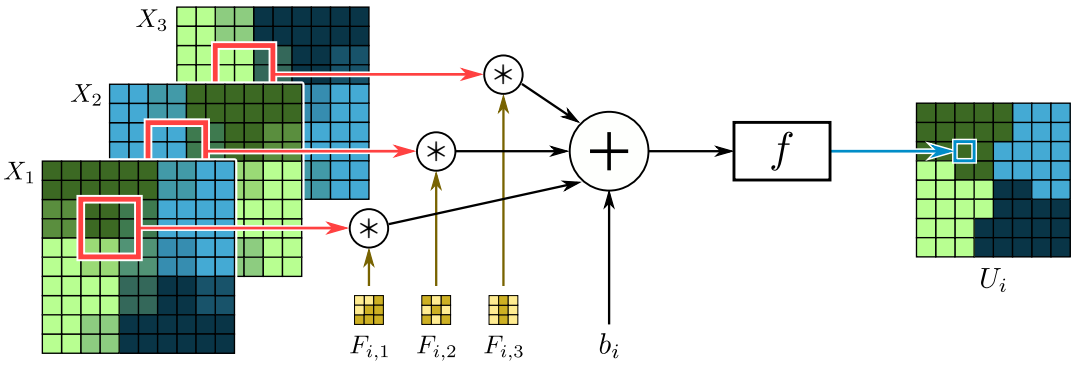


図 2.3: CNN におけるノードの模式図

CNN におけるノードの模式図を図 2.3 に示す．ノード i の出力画像 U_i は，式 (2.1) における入力 x_j を入力画像 X_j，重み $w_{i,j}$ をフィルタ $F_{i,j}$，乗算を畳み込みに置き換えて，以下のように表せる．

$$U_i = f\left(\sum_{j=1}^{n} X_j * F_{i,j} + b_i\right)$$

ただし，行列に対するスカラーの加算は，行列の全要素に等しく同じ値を加算する演算であると定義する．すなわち，バイアス b_i は，畳み込みされた画像の総和 $\sum_{j=1}^{n} X_j * F_{i,j}$ が持つ全要素 (全画素) に対して加算される．また活性化関数 f も同様に，全要素に対して同じ処理を適用する．

処理するタスクによっては，畳み込みを行う層 (畳み込み層) の後に，プーリング層と呼ばれる新たな層を追加する場合がある．プーリング層においては，画像を一定領域ごとに区切り，各画素を 1 画素に置き換えるプーリング (Pooling) と呼ばれる操作を行う．プーリングの手法としては，領域内の最大値を用いる最大値プーリング (Max Pooling)，平均値を用いる平均値プーリング (Average Pooling) 等がある．プーリングの適用により，特徴を小さなサイズに集約できるため，大局的なコンテキストを考慮した推論がしやすくなるほか，画像分類のようなタスクでは，パターンの位置のずれに影響を受けづらくなる (位置不変性が高まる) という利点も得られる．またプーリング層を，フィルタの畳み込みを間隔 (ストライド) を空けながら行うストライド畳み込みに置き換えて，学習可能にする試みも広く行われている．

CNN では，ノード間の結合がフィルタを介した部分的なものになるため，パラメータ数が大幅に少なくなる．そのため，全ての入力に対して重みを介して密に結合している全結合ネットワークと比べ，誤差逆伝播法による学習において誤差の拡散が起こりづらく，より深いネットワークを実現しやすい．

2.5 データ拡張

過学習は，深層学習において大きな問題である．これは，学習データセットに比してネットワークの表現力が高すぎる結果，本質的な特徴をとらえるのでなく，入力と出力の単純な対応関係のみを学習してしまうために起こる．2.3 節で言及したとおり，正則化の導入による改善も図られているが，より根本的には，単純な対応付けでは処理しきれない，多様性に富んだ十分な量の学習データセットを用意することが望ましい．しかしながら，大規模なデータの収集には困難を伴うことが多く，専門家によるアノテーションを要する場合等には，許容できないコストが発生することも考えられる．

そこで，実際にデータを追加することなく，実質的にデータセットの規模を拡大する方法として，データ拡張 (Data Augmentation) が広く用いられている．2 次元画像を例に挙げると，図 2.4 にあるように，水平・垂直反転，任意角度の回転やスキュー，任意倍率の拡大・縮小といったアフィン変換を加える方法がある．

また，カラー画像であれば色情報を変化させることもできる．簡単に行うのであれば，HSV や HSL 色空間で色相，彩度，明度を任意に増減させればよい．ただし，この方法では，色味が大きく変化してしまうため，色が重要な意味を持つタスクでは利用できない．色味を保ちつつ変化を加える場合，YCbCr 色空間へ変換し，輝度 (Y) チャネルのみを増減させる方法のほか，PCA Color Augmentation [27] を利用する方法がある．

PCA Color Augmentation は，主成分分析 (Principal Component Analysis, PCA) に基づいてランダムな色情報の変化を加える手法である．まず，入力された RGB 画像をチャネルごとに分解し，3 つのベクトル X_1, X_2, X_3 を得る．続いて，これらのベクトルから 3×3 の分散共分散行列 $\Sigma_{i,j} = E[(X_i - \mu_i)(X_j - \mu_j)]$ $(1 \leq i \leq 3,\ 1 \leq j \leq 3)$ を求め，Σ

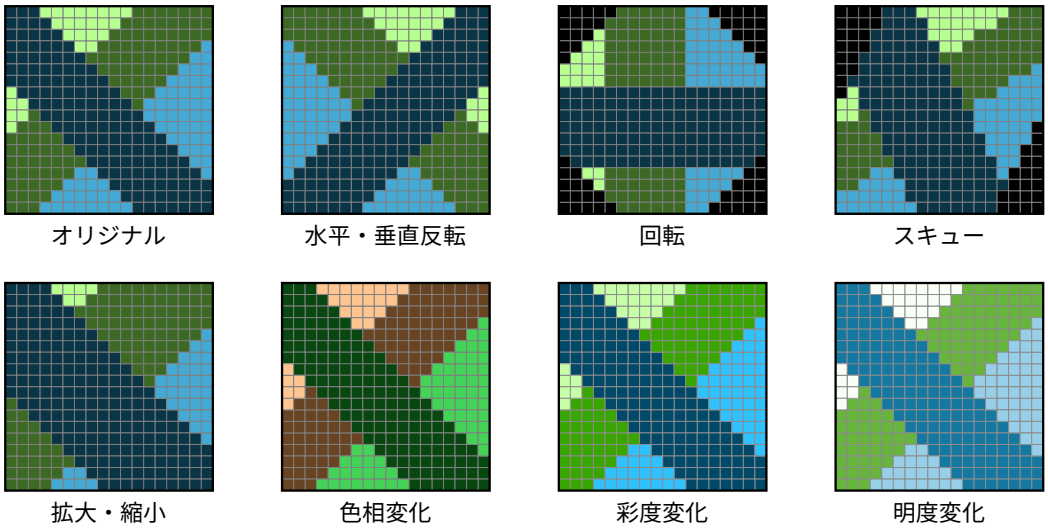


図 2.4: 2 次元画像におけるデータ拡張の例

を固有値分解することで固有値 λ_i と 固有ベクトル $\boldsymbol{p}_i$ を求める $(1 \leq i \leq 3)$．これらの値に基づき，RGB チャネルごとの増減量 $(\Delta_r,\ \Delta_g,\ \Delta_b)$ が以下のようにして得られる．

$$(\Delta_r,\ \Delta_g,\ \Delta_b)^T = (\boldsymbol{p}_1,\ \boldsymbol{p}_2,\ \boldsymbol{p}_3)(\alpha_1\lambda_1,\ \alpha_2\lambda_2,\ \alpha_3\lambda_3)^T$$

ただし，α_i は，正規分布 $N(0, 0.1^2)$ 等からサンプリングされる乱数である．

α_i を変更することで，画像のカラーバランスを大きく損ねることなく，様々なバリエーションを作り出せる．実際に PCA Color Augmentation を適用した画像例を図 2.5 に掲載する．PCA Color Augmentation の考え方は，RGB 画像に留まらず様々な多次元データに応用可能であるため，輝度を用いる方法と比べて汎用性に優れている．

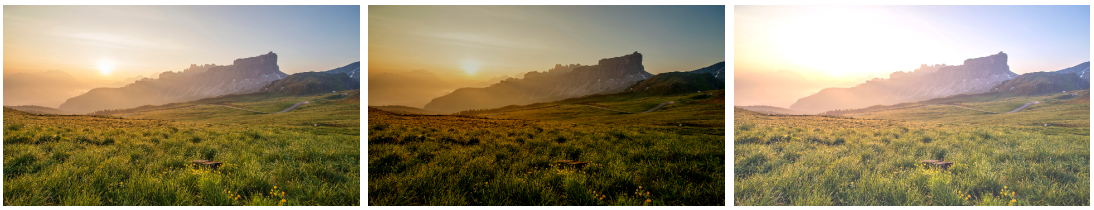


図 2.5: PCA Color Augmentation の適用例

データ拡張は，データセット不足による過学習のリスクを低減する上で有効な手段である．ただし，データ拡張にどのような手法が適しているかは問題によって異なる．例えば，物体認識において色相が重要な役割を果たしている場合，色相を変動させて学習させると性能が低下するおそれがある．また，ある特定の形状に強く依存している場合，スキューのような処理は適切ではない可能性もある．このため，過学習の軽減にあたっては，問題の性質に合ったデータ拡張手法を選定するとともに，ネットワーク構造や学習手法での対策も組み合わせていくことが重要であると言える．

2.6 Residue Number System (RNS)

Residue Number System (RNS) [28] は，数値を複数の法 (Modulus) による剰余 (Residue) の組で表現する仕組みである．RNS は，互いに素な法 m_i から構成される基底 $M = \{m_0,\ m_1,\ ...,\ m_{n-1}\}$ により定義され，全ての法の積 $D = \prod_{i=0}^{n-1} m_i$ が RNS のダイナミックレンジとなる．

RNS では，$0 \leq X \leq D-1$ を満たす任意の整数 X を，剰余の組 $\{x_i \mid x_i = X \bmod m_i\}$ で表せる．この表現の一意性は，中国の剰余定理 (Chinese Remainder Theorem) により保証されている．また，あるしきい値 t $(0 \leq t < D-1)$ を超える X を $X - D$ とみなすことにより，負の数を表現することもできる．この場合，X の範囲は $t-(D-1) \leq X \leq t$ となる．$M = \{2, 5\}$ $(D = 10)$ の場合の例を表 2.1 に示す．

表 2.1: RNS 表現の例

RNS $\{2,\ 5\}$	符号なし	符号あり $(t = 4)$
$(0,\ 0)$	0	0
$(1,\ 1)$	1	1
$(0,\ 2)$	2	2
$(1,\ 3)$	3	3
$(0,\ 4)$	4	4
$(1,\ 0)$	5	-5
$(0,\ 1)$	6	-4
$(1,\ 2)$	7	-3
$(0,\ 3)$	8	-2
$(1,\ 4)$	9	-1

RNS 表現 $\{x_i \mid x_i = X \bmod m_i\}$ から元の整数 X への変換は，全ての法 m_i に対して合同式 $X \equiv x_i \pmod{m_i}$ を満たすような X $(0 \leq X \leq D-1)$ を求めることに他ならない．ここで，以下の 2 条件

$$w_i = p_i \prod_{k \neq i} m_k \ (p_i \in \{0, 1, ..., m_i - 1\})$$
$$w_i \equiv 1 \pmod{m_i}$$

を満たすような重み w_i を用いると，

$$w_k \equiv \begin{cases} 0 \pmod{m_i} & (k \neq i) \\ 1 \pmod{m_i} & (k = i) \end{cases}$$

であることから，$\sum_{k=0}^{n-1} w_k x_k$ は先述した合同式を全て満たし，特殊解の一つであることが分かる．したがって，所望の X は以下の変換式 (2.2) により決定できる．

$$X = \left(\sum_{k=0}^{n-1} w_k x_k \right) \bmod D \tag{2.2}$$

重みの値は，不定方程式 $m_i x + \left(\prod_{k \neq i} m_k\right) y = 1$ を拡張ユークリッド互除法により解く ($y \bmod m_i = p_i$)，あるいは p_i を総当たりにより決定することで求められる．重みを事前に計算しておくことにより，RNS から整数への変換は，重み付き総和のダイナミックレンジによる剰余をとることで簡単に行える．例として，$M = \{2, 5\}$ に対する重みは $\{1 \times 5,\ 3 \times 2\} = \{5, 6\}$ であるから，

$$(1, 2)_{RNS} = (5 \times 1 + 6 \times 2) \bmod 10 = 17 \bmod 10 = 7$$

となる．

RNS における加算，減算及び乗算は，要素ごとのモジュロ演算により行う．$M = \{2, 5\}$ の符号付き RNS ($t = 4$) における演算例を以下に挙げる．

$$\begin{aligned}
3 + (-5) &= (1, 3)_{RNS} + (1, 0)_{RNS} \\
&= ((1 + 1) \bmod 2,\ (3 + 0) \bmod 5)_{RNS} = (0, 3)_{RNS} = -2 \\
-1 - (-4) &= (1, 4)_{RNS} - (0, 1)_{RNS} \\
&= ((1 - 0) \bmod 2,\ (4 - 1) \bmod 5)_{RNS} = (1, 3)_{RNS} = 3 \\
2 \times (-2) &= (0, 2)_{RNS} \times (0, 3)_{RNS} \\
&= ((0 \times 0) \bmod 2,\ (2 \times 3) \bmod 5)_{RNS} = (0, 1)_{RNS} = -4
\end{aligned}$$

このように，RNS を用いることで，大きな数同士の加算及び乗算を，複数の簡単なモジュロ演算に分解できる．これらの演算は，基底 M を小さな整数で構成することにより，桁上がりを意識することなくルックアップテーブル (LUT) を用いて効率良く実現できるため，ハードウェア実装に適している．

RNS の問題点としては，除算及び大小比較が容易に行えないということが挙げられる．ただし，畳み込みニューラルネットワークへの応用を考えた場合，演算の大部分は畳み込み演算 (乗算及び加算) であり，除算や大小比較を必要とする活性化関数の適用等は，畳み込み後の少数の値に対してのみ行われるため，一旦整数表現に戻してから演算しても影響は小さく抑えられると考えられる．

また，法が互いに素でなければならないことに起因して生じる，演算の粒度が均一とならない点，あるいは，大きなダイナミックレンジが必要な場合に，法として大きな値を使わざるを得ない点も問題である．これらの解決策として，大きな法による剰余を，複数のより小さな法を用いてさらに分解する Nested RNS [29] も提案されている．

分解にあたっては，想定される演算に基づいて十分なダイナミックレンジを確保しなければならない．例として，法 17 を分解する場合を考える．RNS 上で，2×2 の画像とフィルタの畳み込みを行う場合，積 4 個の加算が行われるため，必要なダイナミックレンジは $17^2 \times 4 = 1156$ である．したがって，法 17 は，より小さい法より成る基底 $\{3, 5, 7, 13\}$ ($D = 1365$) 等を用いて分解できることが分かる．分解に用いる法が大きくなりすぎた場合は，さらに分解を繰り返していく．

Nested RNS では，基底の構成が複雑となるものの，小さな法を何度も使い回すことができるため，設計によっては，演算器の共有により資源使用量の削減につながる．

第3章

リアルタイム超解像システムの実装と評価

本章では，畳み込みニューラルネットワークを用いたリアルタイム動画像超解像システムの設計及び FPGA への実装を示し，その性能を評価する．

3.1 背景と目的

近年，高解像度のディスプレイが急速に普及している．例えば，スマートフォンやラップトップコンピュータ等のモバイルデバイスにおいては，すでにフル HD (1920×1080) 解像度が一般的となり，WQHD (2560×1440) やそれ以上の解像度も広く用いられるようになっている．また，テレビジョンや PC モニタといった中大型ディスプレイにおいては 4K UHD (3840×2160) 解像度が普及を始めており，8K UHD (7680×4320) へ向けた研究開発も行われている [30, 31]．これに伴い，それらのディスプレイ上に表示されるべき画像に対しても，従来より高い解像度が要求されている．

しかしながら高解像度の画像，とりわけ動画像を扱うにあたっては，いくつかの技術的な問題が存在する．イメージセンサやディスプレイといった入出力機器の製造難度が高まることはもちろんのこと，単位時間あたりに扱う画素数が膨大になることから，データの符号化及び復号が複雑なものとなり，H.265/HEVC [32] といった最新の非可逆圧縮技術をもってしてもビットレートが高くなることから，データの保存や伝送にも困難を伴う．とりわけ，モバイルネットワークを通じた高解像度動画の配信は，トラフィックの観点から厳しい制約を受けることとなる．2021 年までに，モバイルネットワークの総トラフィックに占める動画データの割合が 78% に達するとの試算もあり [33]，電波資源に限りがある中で，高ビットレートの動画配信が急速に広がることは，技術的な困難に留まらず，ネットワークの中立性を脅かすといった倫理的問題にもつながり得る．将来的に移行が見込まれる 8K UHD フォーマットでは，解像度の高さに加え，ビット深度やフレームレートの更なる拡充も行われるため，この問題は一層深刻なものになると予想される．

また，過去に製作された低解像度の画像コンテンツを，高解像度のディスプレイで鮮明に表示することも重要である．例えば，HD (1280×720) 解像度の画像を 4K UHD ディスプレイに表示する場合，縦横各 3 倍，画素数にして 9 倍の拡大が必要であり，バイリニア法やバイキュービック法といったシンプルなアルゴリズム [34] で補間するだけでは十分な品質が得られない．

これらの問題に対する解決策の一つとして，低解像度画像から高解像度画像を再構成する超解像 (Super-Resolution) 技術がある．失われた情報を補間するアルゴリズムには様々

なものが提案されている [35, 36] が，近年提唱された手法として，畳み込みニューラルネットワークを用いた超解像がある [11, 12, 13, 14]．低解像度画像と高解像度画像の関連性をネットワークに学習させることにより，高い品質の超解像を実現できるが，既存研究の多くはハイパフォーマンス GPU を用いた実装であることから，消費電力が大きく，組み込み機器での可用性には制限がある．

そこで本研究では，畳み込みニューラルネットワークを用いたリアルタイム動画像超解像システムを設計し，FPGA に実装することとした．

3.2　関連研究

本節では，超解像技術を扱った関連研究について，畳み込みニューラルネットワークを用いたものを中心にまとめる．

3.2.1　畳み込みニューラルネットワークを用いない手法

W. T. Freeman らは，低解像度のパッチ (小画像) と，対応する高解像度のパッチとの組で構成されたデータベースを基に，入力画像の各領域に対して低解像度パッチとのマッチングを行い，高解像度パッチに置き換えることで超解像を行う手法を提案している [35]．データベースが適切に構築されていれば，完全に失われたディテールを比較的自然に補うことができるが，対象とする画像のサイズや，非可逆圧縮の適用状況等に応じてデータベースを構築し直さなければならず，汎用性に乏しいという欠点がある．また，高い品質を得るためには大規模なデータベースが必要であり，探索に要する演算量やメモリ使用量が増大する点も問題である．

S. C. Park らにより発表された論文 [36] では，同一対象物を異なる位置あるいは時刻において撮影した複数の画像を用いて，それらの画素値を，サブピクセル精度の動き検出に基づいて超解像画像にマッピングすることにより超解像を行う手法が提案されている．1 枚の画像から処理を行う場合と比べ，より精度の高い超解像を実現できる利点があるものの，超解像の適用対象は主に動画に限られる．また，複数枚の画像を保持することによる大きなメモリ使用量のほか，広範囲かつ高精度の動き検出を行うことによる演算量の増大も課題となる．加えて，オクルージョン等の要因により適切な対応点がそもそも存在しない場合，正しい結果が得られないことも考えられる．

3.2.2　事前拡大手法

畳み込みニューラルネットワークを超解像に適用する場合，学習において，訓練データに低解像度画像，教師データに高解像度画像を用い，低解像度画像から高解像度画像を推定するようにネットワークを訓練することが基本的な考え方となる．しかしながら，低解像度空間から高解像度空間へのマッピング手法や，誤差の評価手法等により，システムの構成には様々なバリエーションがあり得る．

C. Dong らにより発表された論文 [11, 12] では，3 層の畳み込みニューラルネットワークを用いた静止画像の超解像システムが提案されている．提案されたモデルは Super-Resolution

Convolutional Neural Network (SRCNN) と命名されており，複数の客観的画質評価指標に基づいて既存の超解像アルゴリズムと比較し，処理時間と超解像品質のバランスに優れていることが示されている．

SRCNN においては，事前に入力画像を，バイキュービック補間により高解像度画像と同じサイズまで拡大し，これを学習済みのネットワークに与え，エッジやテクスチャ等の補間を行い，高解像度化された画像を得るという仕組みを採用している．なお，この事前拡大は，転置畳み込み (Transposed Convolution) を用いることにより，ネットワーク内部に組み込むこともできる．転置畳み込みの詳細については，4.3.5 項を参照されたい．図 3.1 は，SRCNN の事前拡大に基づく手法の概念を表したものである．しかしながらこの手法には，ネットワークが実質的に利用できる情報量が小さいという問題がある．

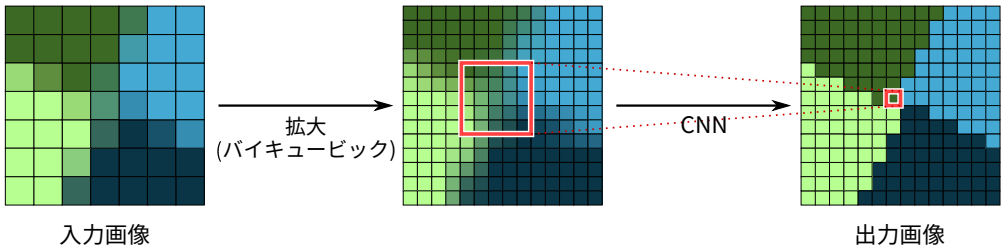


図 3.1: 事前拡大手法の概念図

[12] に示された基本構成では，各層のフィルタサイズが入力層から順に 9×9，1×1，5×5 であるから，出力画像の 1 画素は入力画像の 13×13 画素から生成されることになる．しかし，入力画像は事前に拡大されているため，例えば縦横 2 倍の超解像を適用する場合，実質的には 6.5×6.5 画素相当，つまり本来期待される情報量の 25% しか利用されない．超解像スケールが大きくなれば，この乖離はより顕著となる．これは，資源制約等の理由によりフィルタサイズを大きくできない状況においては特に大きな問題となると考えられる．例えば，フィルタサイズが 3×3 である場合，畳み込みにおいて利用される情報量は 1.5×1.5 画素相当と，ごく限られたものとなる．

また SRCNN は，既存のいくつかの超解像アルゴリズムと比較し，同一品質であればより高速に処理できることが示されているが，静止画を対象としており，[12] におけるソフトウェア実装では，構成に応じて数秒程度の処理時間を要する．そのため，動画像をリアルタイムに高解像度化するという用途には用いることができない．

SRCNN は順伝播型の畳み込みニューラルネットワークであるが，J. Kim らは，再帰的なネットワーク構造を導入した超解像システムである DRCN (Deeply-Recursive Convolutional Network) を提案している [37]．このシステムは，同一の畳み込み層を再帰的に繰り返し適用する仕組みにより，パラメータの数を増やすことなく品質を改善している．著者らはまた，勾配消失や爆発を軽減して学習の困難性を軽減するための訓練技法についても提案している．

DRCN は SRCNN よりも優れた品質を実現したが，リアルタイムの超解像を実現するために FPGA でパイプライン設計を行う場合，多数回再帰適用される畳み込み層を複製して空間展開しなければならず，また，利用可能な資源量の関係上，パラメータ数にも厳しい制約があることから，パラメータを増やさずに済む利点がそもそも活かしづらい．

3.2.3　サブピクセル再構成手法

W. Shi らは，SRCNN のように事前拡大を用いず，低解像度空間で処理を行い，品質の向上を実現した超解像システムを提案し，Efficient Sub-Pixel Convolutional Neural Network (ESPCN) と呼称している [13].

ESPCN におけるサブピクセル再構成手法の概念を図に表すと，図 3.2 のようになる．この手法では，事前拡大を用いて高解像度空間への変換を行う代わりに，出力層に．サブピクセルレベルで異なる位置に対応する複数のチャネルを用意し，それらから得られる複数枚の出力を 1 枚の画像に再構成することにより超解像を行う．この再構成手法は，Pixel Shuffler とも呼ばれる．例えば，縦横 2 倍の超解像を行う場合，図 3.2 に示されているように，$2 \times 2 = 4$ 枚の画像を格子状に組み上げて最終的な出力画像を得る．同様に，縦横 n 倍の超解像では，n^2 枚の出力を利用することとなる．

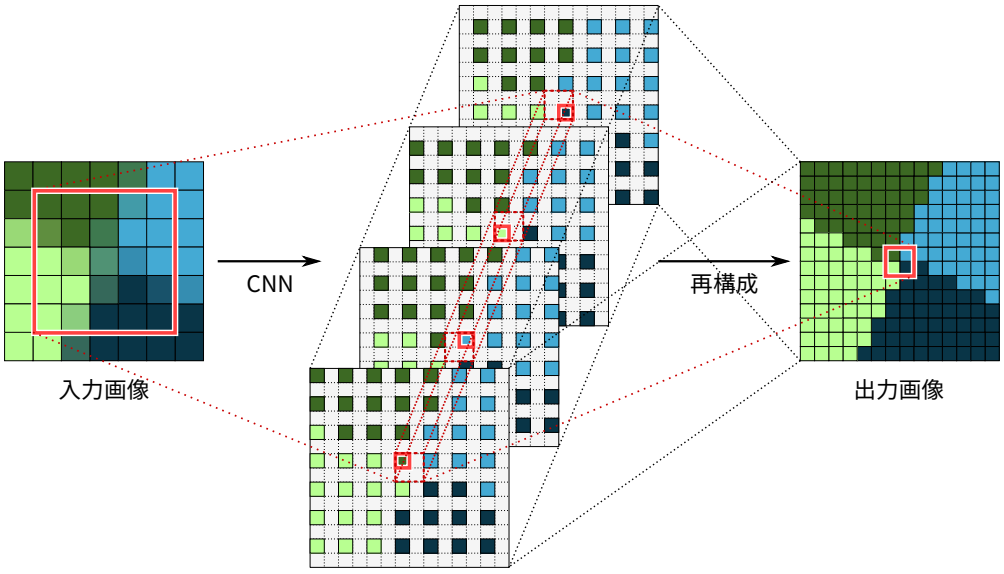


図 3.2: サブピクセル再構成手法の概念図

この仕組みにより，事前拡大を行うことなく，低解像度画像をそのままネットワークに与えることができる．これにより最大限の情報を利用した超解像が可能となり，事前拡大やそれに類する手法により高解像度空間で処理を行う場合と比べ，より高い品質を実現できることが示されている．また，ネットワークの演算が低解像度空間で行われるため，ネットワーク全体の規模を同一に抑えた場合，事前拡大手法に対して計算量が $1/2^n$ に削減される．これにより，ハイパフォーマンス GPU を用いることで，リアルタイムでの動画像処理も可能となっている．

一方で，出力層が複数のチャネルを持つことから，上位層の構成が同じ場合はネットワークの規模が大きくなり，計算量が増加する．また，上位層では複数のチャネル全ての出力に対応できるような特徴を抽出していく必要があるため，1 画素の出力に特化する場合と比べて，より高い表現力が求められる，つまりより大規模なネットワークが必要になると考えられる．

J. Caballero らは，動画像の超解像をターゲットとした ESPCN の派生システム VESPCN (Video ESPCN) を提案している．VESPCN では，動き補償 (Motion Compensation) を用いることにより，空間方向だけでなく時間方向の情報も組み合わせて処理を行う．これにより，1 枚絵を用いる場合と比べ，より多くの情報を考慮できることから，より高い品質を達成できる．しかし，フレームバッファのために多量のメモリが必要になること，動き補償のために追加の計算コストが生じることが欠点である．

3.2.4　Generative Adversarial Network を用いた手法

C. Ledig らによって提案された SRGAN [14] は，Generative Adversarial Network (GAN) [1] を応用した超解像システムである．

GAN は，ニューラルネットワークのモデルの一つで，Generator 及び Discriminator と呼ばれる 2 種類のネットワークから成り立つ．Generator は，ノイズを入力として教師データらしい出力を生成するためのネットワークであり，Discriminator は，入力データが単体で与えられたとき，それが教師データなのか，あるいは Generator が生成したものなのかを見分けるためのネットワークである．両者が競い合うように学習を進めることにより，最終的に Generator は，単体で見ると教師データと区別がつかないようなデータを生成するようになることが期待される．

畳み込みニューラルネットワークを用いた一般的な超解像では，ネットワークの出力画像と教師画像の平均誤差を最小化するように学習を進めていくため，無数に存在し得る解のうち，いずれに対しても誤差が平均的に小さくなるもの，つまりややぼやけた画像を生成するようになる傾向がある．これを図示すると，図 3.3 のようになる．解の候補として示した画像群は，いずれも面積平均法で縮小すると入力画像と等しくなるため，一意に解を決定することは不可能である．そこで，平均誤差ベースの超解像では，そのいずれに対しても極端な差のない出力を生成する．これに対して SRGAN は，Discriminator が教師画像かどうか判別できなければ十分であるため，図 3.3 の例のように，無難な解にとらわれず，より鮮明な処理結果が得られるという特徴を持つ．

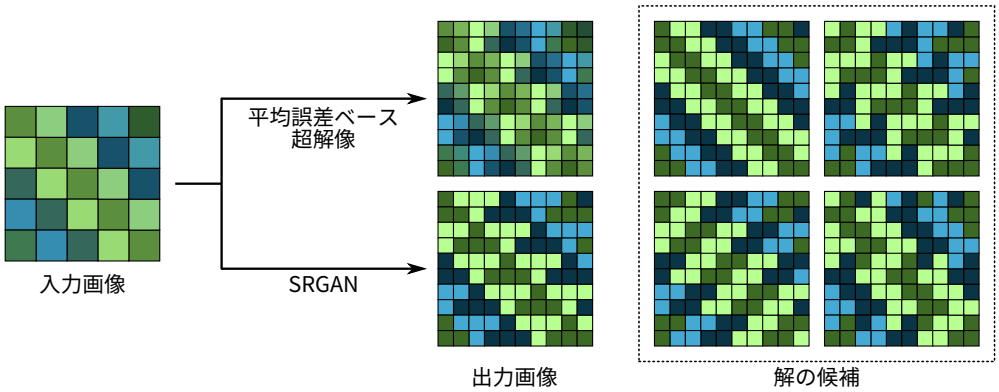


図 3.3: 平均誤差ベースの超解像と SRGAN の違い

ただし，入力画像に対応する真の高解像度画像に近いかどうかではなく，あくまでも高

解像度画像らしいかどうかが重視されるため，本来のものとは全く異なるテクスチャが生成されてしまうことが欠点である．それに伴い，客観的画質評価指標に基づく評価でもバイキュービック補間を下回る場合があり，忠実性という観点からは必ずしも優れた性能を発揮しない．例えば，監視カメラ画像の鮮鋭化といった用途には不向きであるし，映画等の芸術作品への適用に際しても，製作者の意図と乖離した出力がなされることが懸念されるため，応用範囲が限定されると考えられる．また，GAN 全般の問題であるが，ネットワークの学習が不安定である点も欠点として挙げられる．

なお，SRGAN における高解像度画像の生成は，ESPCN と同様のサブピクセル再構成手法に基づく．したがって，事前拡大手法と比較しより多くの情報が扱え，パフォーマンスにも優れるという利点はそのまま引き継がれている．

3.3　設計

本節では，3.2 節で扱った関連研究を踏まえ，本研究における超解像システムの設計について，その特徴的な要素を中心として概説する．実装の詳細については，3.4 節にて述べることとする．

3.3.1　設計指針

3.2 節で述べたとおり，超解像を実現する手法には様々なものがあるが，本研究においては，近年画像認識等の分野で大きな成果をもたらし，超解像においても優れた性能を示す畳み込みニューラルネットワークを採用することとした．ニューラルネットワークにおいては，データが持つ規則性の学習が適切に行われれば，未知のデータに対しても比較的良い結果を出すことができる汎化性が得られる．これは，データベースを用いた手法に見られる欠点を解決し得ると考えられる．

畳み込みニューラルネットワークでは，学習時，推論時ともに，並列性の高い大量の積和演算を行うことから，一般的には GPU を用いて処理を行うことが多い．しかしながら，GPU には単体動作を行うための十分な命令制御機構が備わっていないため，多くの場合，CPU 上で別途プログラムを動作させ，そこから適宜 GPU に演算を発行するという形態を採る．また，高パフォーマンス GPU は一般に消費電力も大きい．これらの理由により，組み込み系の機器での可用性には制限がある．また，GPU 処理を行う場合，外部から入力された映像はまず CPU 側のメインメモリに格納され，そこから GPU へ転送されて処理された後，再び CPU 側に転送されてから I/O を通じて出力されるという手順を踏むため，データの移動に伴う消費電力およびレイテンシの増大も避けられない．

そこで本研究では，より広範な機器で利用でき，電力効率にも優れたシステムを構築可能な FPGA を利用し，ネットワークをフルパイプラインで構築することで，低レイテンシでのリアルタイム処理を実現する．

3.3.2　反転手法

今回設計したシステムでは，3.2.2 項において述べた事前拡大に起因する問題を解決するため，拡大の代わりに水平及び垂直方向の反転を組み合わせて適用する手法を導入した．図 3.4 に，この反転手法の概念図を示す．

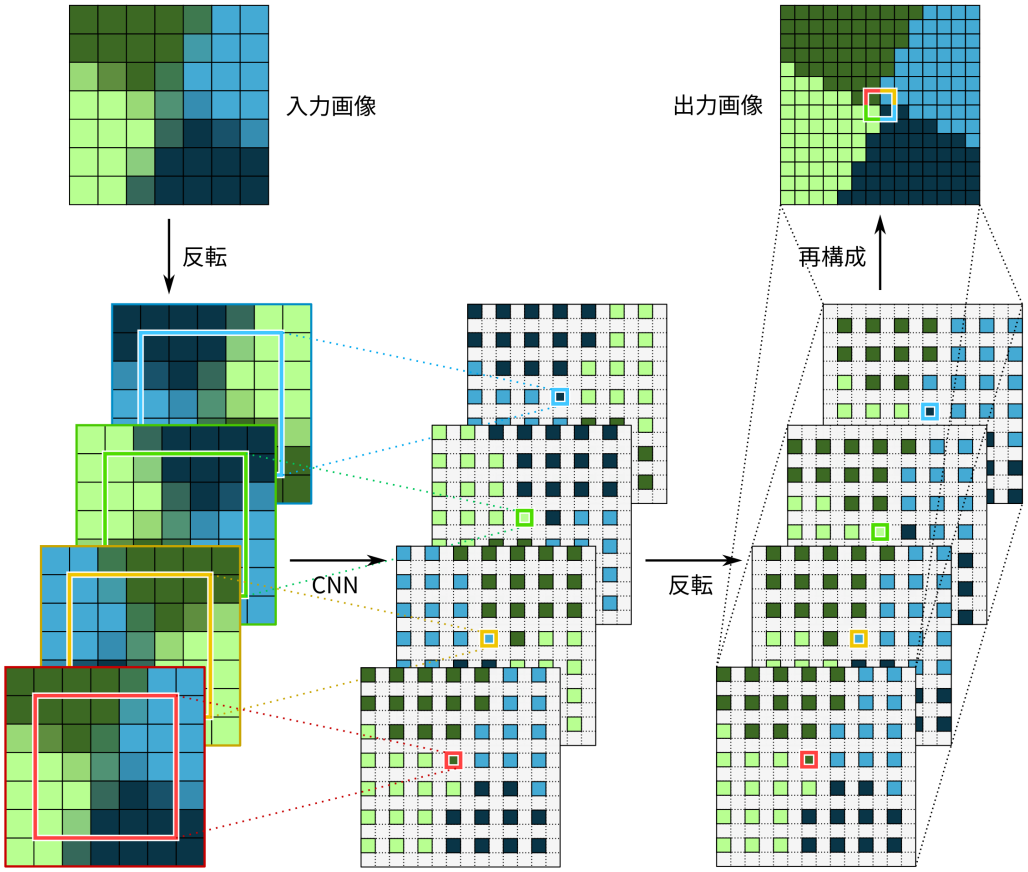


図 3.4: 反転手法の概念図

畳み込みニューラルネットワークによる超解像では，入力画像中の小領域を基にして出力画像の各画素を復元していく．この小領域をパッチと呼ぶこととする．パッチの大きさは，ネットワークの深さと各層のフィルタサイズに基づき決定されるが，今回の設計では FPGA の資源制約により，9×9 という比較的小さなサイズを用いることとなる．

本システムは，2×2 倍の超解像を行うように設計されている．SRCNN に見られるような事前拡大を用いる手法では，入力画像をあらかじめ 2×2 倍に拡大し，出力画像と同じサイズにしておくことで，図 3.1 に示されているように，出力画像の各画素を異なるパッチに対応付けることができる．一方，事前拡大を行わない場合，入力空間の大きさが出力の 25% しかないため，何かしらの工夫がなければ，平均して出力の 2×2 画素を単一のパッチに割り付けることにならざるを得ない．

3.2.3 項で示したサブピクセル再構成手法では，単一のパッチから 4 画素を出力するようにネットワークを構成することで，この問題に対処している．一方，本研究の反転手法で

は，あるパッチに本来対応付けられる 4 画素のうち，左上の画素のみを復元するようネットワークの学習を行い，残りの 3 画素については，水平及び垂直反転を組み合わせて復元を行う．例えば，あらかじめパッチに水平反転を適用しておくと，ネットワークは，反転後のパッチに対して左上にあたる画素を出力するが，これは反転前のパッチに対する右上の画素に相当する．つまり，事前に水平反転を加えてネットワークに与えることで，右上の画素を復元できたことになる．同様の仕組みにより，垂直反転を加えることで左下の画素を，両方向の反転を併用することにより右下の画素を復元できる．

このように，反転手法では 4 画素の位置関係を利用することで，ネットワークに拡張を加えることなく，低解像度空間での超解像処理を可能にする．事前拡大手法では，パッチが持つ実質的な情報量は，パッチサイズの 25% 相当に限られる．一方の反転手法では，パッチサイズを最大限に活かした超解像を行うことができる．また，同じく事前拡大を要しないサブピクセル再構成手法と比較しても，反転手法では左上画素のみに特化できるため，より小さいネットワークでも対応しやすい利点がある．反転手法による品質の向上については 3.6.2 項で述べることとする．

なお，反転手法において超解像スケールは原理上 2×2 に限定されるが，求める出力サイズ以上になるまで超解像処理を繰り返し，必要に応じて何らかの縮小アルゴリズムで調整することで，任意のスケールに対応できる．

3.3.3　ネットワーク構成と活性化関数

本システムでは，リアルタイム処理を実現するために，ネットワークの全演算を空間展開してパイプラインを構築し，ストリーム処理を行うこととしている．そのため，FPGA で利用可能な資源の量を考慮しながら，ネットワーク構成を決定しなければならない．

扱う画像は RGB 8 ビットのフルカラー画像とする．そのため，入出力のチャネル数は 3 である．人間の視覚において，色解像度は輝度解像度と比べて大きく劣ることから，扱うチャネルを輝度のみに限定する手法もある．カラー画像を扱う場合，YCbCr 色空間に変換した上で輝度信号 (Y) のみに超解像を適用し，色差信号 (Cb, Cr) については，バイキュービック補間等で拡大した上で合成すれば良い．関連研究 [12] において，各種色空間における超解像の効果を検討した結果，輝度のみを扱う場合でも品質は大きく劣化しないことが示されている．

しかし，輝度と色差を別々に扱う都合上，エッジに偽色が生じることは避けられない．また，パイプライン実装を行う場合，Cb, Cr の信号を蓄積するバッファを別途設けなければならないので，設計が複雑化する．これらを考慮し，今回，RGB の色空間を直接扱う設計としている．3.6.2 項で示すように，輝度のみを扱った従来の設計 [38] と比較して，同程度のパラメータ数で同等以上の品質を実現できる．

各層のフィルタサイズは，畳み込みニューラルネットワークで一般的に用いられている 3×3 に統一し，ネットワークの深さは 4 層とする．ただし，この層数は畳み込み層の数に基づくものであり，入力層は層数に含めていない．したがって，隠れ層は 3 層となる．学習時に誤差逆伝播が効率良く行われるよう，各層には ResNet [23] に見られるショートカット接続を導入している．

フィルタサイズ 3×3 で 4 層のネットワークの場合，3.3.2 項で述べたパッチサイズは

9×9 となる．事前拡大は行わないため，出力画像の各画素は，入力画像にある 9×9 の小領域から復元されることになる．このような小領域は，一般には受容野 (Receptive Field) とも呼ばれ，大きいほど，広い範囲のコンテキストを考慮した推論が行われることを示す．ネットワークの構成と，パイプラインが流れる様子を，図 3.5 に示す．

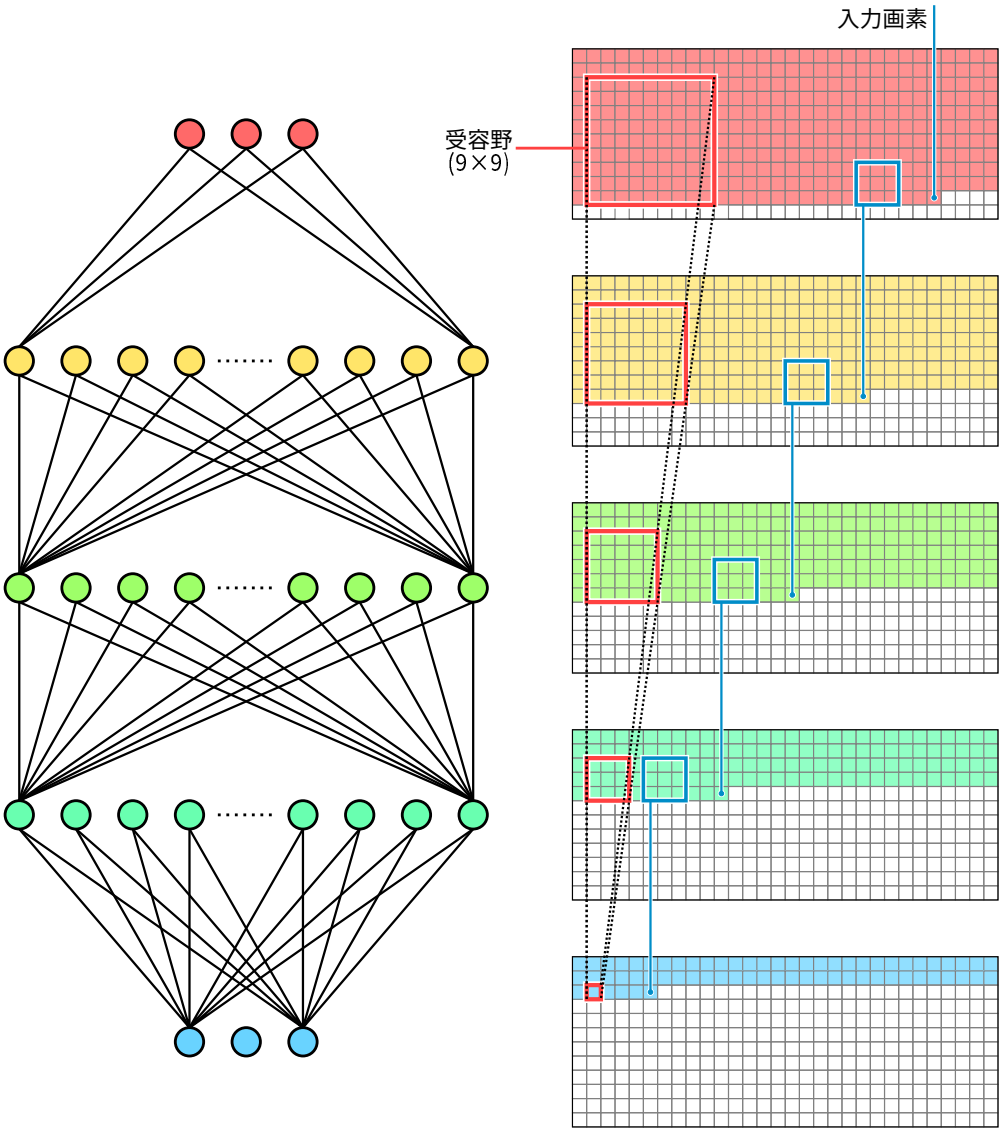


図 3.5: ネットワーク構成とパイプラインの流れ

ネットワークの活性化関数には，Leaky ReLU [21] $f(x) = \max(ax,\ x)\ \ (0 < a < 1)$ を用いる．一般的に用いられている ReLU [19] $f(x) = \max(0,\ x)$ では，いかなる負の入力に対しても 0 が出力されることから，学習が進むうちに，計算過程で絶対値が大きい負数が出現するようになることがある．本システムは，3.3.4 項で述べるように固定小数点演算を採用しているため，この現象は必要なビット幅の増大，ひいては資源使用量の増加に

つながる．一方 Leaky ReLU は，負領域においても 0 でない重み a を持つため，この問題を緩和できると期待される．両活性化関数の比較については 3.6.3 項にて行う．本システムでは，a の値を $2^{-2} = 0.25$ とした．これにより，2 ビットの右算術シフトをベースとした簡単な実装を可能にしている．

3.3.4　RNS に基づく演算

2.6 節で述べたように，数値表現として RNS を用いることで，加算及び乗算を，LUT により効率良く実装できる．そこで本研究では，資源使用量を削減するため，加算及び乗算を主たる演算とする畳み込みニューラルネットワーク内部に RNS を採用した．なお，Nested RNS [29] については，パイプラインに演算を空間展開する，つまり演算器の使い回しをしない本システムでは効果が見込めないこと，また，パラメータ (フィルタ係数及びバイアス) に全て固定値を用いることにより，乗算を単なる 1 値入力の変換テーブルに置き換えられるため，比較的大きな法を利用でき，通常の RNS でも十分なダイナミックレンジが確保できることから採用していない．

演算そのものは，資源使用量を削減するため固定小数点演算により行う．ネットワークの各層において，入力された 2 進数固定小数点数 X が，n ビットの小数部を持つものとする．このとき，X を 2^n 倍することにより整数とみなしたもの，つまり $2^n X$ が RNS 表現に変換される．続いて，同じく 2^n 倍された上で RNS 表現に変換されたフィルタ係数 $2^n F$ との乗算が行われ，積 $2^{2n} XF$ を得る．これは，小数部が $2n$ ビットに増加したことと等価であるが，RNS における除算の困難性から，ビットの切り詰めは行われず，そのまま積の総和計算，2^{2n} 倍されたバイアス $2^{2n} B$ の加算までが行われる．

以上のようにして得られた処理結果は，再び 2 進数固定小数点数に戻され，ビット切り詰めや活性化関数の適用等，除算 (ビットシフト) や大小比較を必要とする処理が行われた上で次の層に渡されることとなる．

ある層の入力数 (前層の出力数) が i，出力数 (ノード数) が j，フィルタ画素数が k であるとき，パイプラインに展開される乗算回数は ijk，加算回数は $ijk - 1$ であるのに対し，2 進数から RNS への変換は i 回，RNS から 2 進数への変換は j 回のみであるため，総演算量への影響は限定的である．

3.4　実装

本節では，本研究における超解像システムの実装について，3.3 節で述べた設計の概要を踏まえながら，モジュール単位で説明する．システムは全て，ハードウェア記述言語である SystemVerilog を用いてレジスタ転送レベル (Register Transfer Level, RTL) で設計されており，全体の構成を概観すると図 3.6 のようになる．

3.4.1　入力

本システムは，外部から，入力画像の RGB 画素値 (in_r，in_g，in_b) が，ラスタスキャンにより座標 (in_vcnt, in_hcnt) とともに与えられることを前提に設計されている．なお，

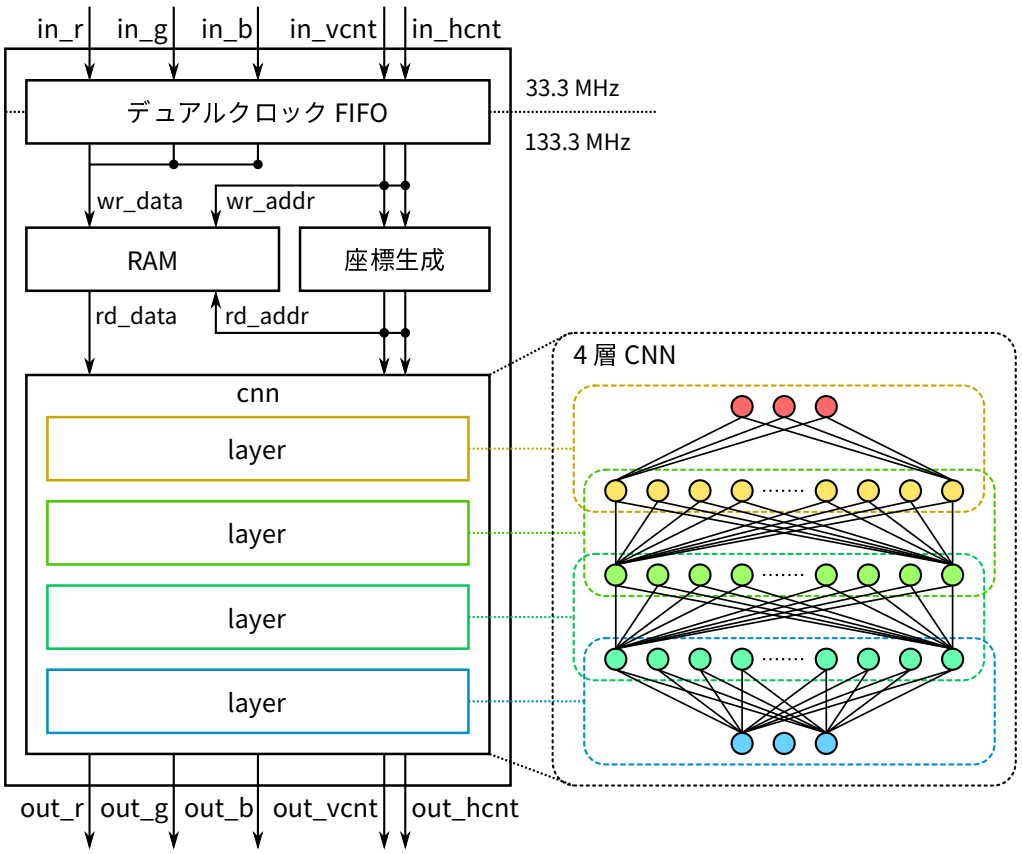


図 3.6: システムの構成図

画素値は 8 ビットの符号なし整数であり，in_vcnt，in_hcnt はそれぞれ，画像の左上を原点としたときの垂直及び水平座標である．入力画像のサイズは Quarter HD (960×540) であり，超解像が適用された出力画像のサイズは，その縦横 2 倍であるフル HD (1920×1080) となる．なお，同期領域はサイズに含めていない．フレームレートは 60 fps を想定しているが，回路の最大動作周波数の範囲内であれば任意の値に対応可能である．

3.4.2　画素ストリームのインタリーブ

3.3.2 節で述べたように，本システムで採用する反転手法では，反転の種類に応じた 4 種類の入力画像を基に超解像を行うが，それぞれの画像に対して独立したパイプラインを構成するとシステムが複雑化する．そこで今回の実装では，4 画像を 1 ストリームにインタリーブすることとした．また反転そのものは，入力時ではなく，ネットワークの各層で畳み込みのたびに行うため，ここでは単に入力の 1 画素を $2 \times 2 = 4$ 回出力するだけで良い．つまり，ニアレストネイバー法による拡大と等価である．したがって，反転のための大きなフレームバッファ等は不要で，レイテンシの増大もない．このインタリーブの様子を図示したものが図 3.7 である．また，反転の種別は，表 3.1 に示すように，垂直座標 v と水平座標 h それぞれの偶奇の組み合わせに対応している．

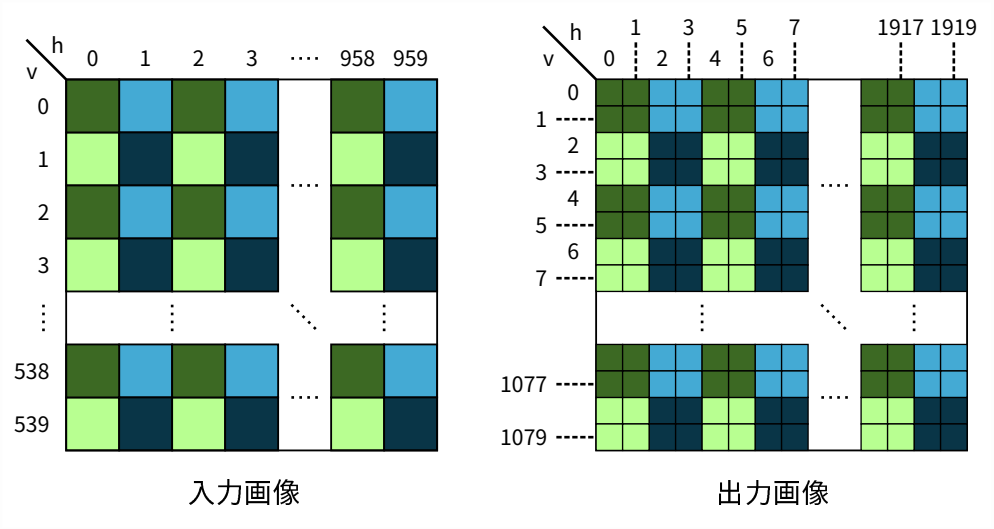


図 3.7: 各反転種別に対応する画素のインタリーブ

表 3.1: インタリーブにおける座標と反転種別の対応関係

$v \bmod 2$	$h \bmod 2$	反転種別
0	0	反転なし
0	1	水平反転
1	0	垂直反転
1	1	水平・垂直反転

インタリーブ後は画素数が 4 倍になるため，パイプラインを駆動するクロック周波数はピクセルクロックの 4 倍となる．そのため，図 3.6 にあるように，クロックドメイン間で画素値や座標データを受け渡すためにデュアルクロック FIFO を用いている．渡された画素値 (計 24 ビット) は，座標に応じて RAM に格納される．また，出力の基準となる座標を，入力座標を $(v,\ h)$ としたとき，$(2v,\ 2h)$ をベースとし，RAM に十分な画素が保存されるまでに必要なレイテンシを考慮して生成する．生成された出力座標をもとに，RAM から対応する画素値を読み出すことで，インタリーブされた画素ストリームを生成する．具体的には，生成された出力側の座標を $(v,\ h)$ とすると，以下の式のように表される．

$$O(v,\ h) = I(\lfloor v/2 \rfloor,\ \lfloor h/2 \rfloor)$$

ただし，O はインタリーブ後の画像，I は入力画像である．生成された画素ストリームと座標は，ネットワーク本体である後段の *cnn* モジュールに渡される．

3.4.3　*cnn* モジュール

cnn モジュールは，反転手法に基づき入力画像ストリームに超解像を適用する，4 層の畳み込みニューラルネットワークである．図 3.6 に示したように，ネットワークの各層に

対応する *layer* モジュール群を内包している．本モジュール及びそのサブモジュール群は全てパイプライン化され，毎クロック 1 画素を入力し，ストリーム処理により，毎クロック 1 画素を出力するよう設計されている．

本モジュールの入力は，インタリーブされた RGB 画素値のストリームである．まず，入力された 8 ビット整数の各画素値を，$n(\geq 8)$ ビットの小数部を持つ 0 から 1 までの固定小数点数に変換する．続いて 4 つの *layer* モジュールを通して超解像を適用し，復元された画素値を再び 8 ビット整数に変換して，座標とともに出力する．なお，固定小数点画素値 x の整数への変換は，以下のように，最近値への丸めによって行う．

$$\mathrm{int}(x) = \min(\max(0, \lfloor 2^n x + 0.5 \rfloor), 255)$$

入力，出力及び重みの小数部ビット幅や，フィルタサイズ，ノード数等は，*layer* モジュールのパラメータとして，層ごとに独立して指定できるよう設計されているが，今回の実装では，小数部ビット幅は層や対象によらずネットワーク全体で一定とし，フィルタサイズも 3×3 に統一している．

3.4.4　*layer* モジュール

layer モジュールは，畳み込みニューラルネットワークの各畳み込み層に相当する役割を担う．図 3.8 にモジュールの構成を示す．

本モジュールは，前層の各ノードが出力した，小数部 n ビットの固定小数点数である画素群を入力とする．そのそれぞれについて，まず，3.3.4 項で述べたように，2^n 倍することにより整数とみなして，*rns_encoder* モジュールにより RNS 表現への変換を行う．RNS 表現はひとまとまりのビット列にパックされ，RNS 専用の加算器及び乗算器のモジュールを用意することにより，固定小数点数と同様に扱えるように設計している．

続いて，変換された RNS ストリームそれぞれについて，シフトレジスタと FIFO を用いて実装された移動ウインドウで，フィルタと同サイズ，つまり 3×3 の小画像を得る．3.4.2 項で述べたインタリーブのため，ここでは 1 画素ずつ間隔を空けながら画素を取り出す必要があることに注意されたい．FIFO はデュアルポート RAM を用いた設計となっており，フィルタサイズが $k \times k$ である場合，計 $(2k-2)$ 個の FIFO が必要となる．なお，小画像の中央にあたる座標が入力画像の最外周部に位置する場合，得られた小画像には，入力画像の範囲外にあたる無効な画素が含まれることになる．そのような場合には，最も近い有効な画素を拡張して無効な画素を埋めるパディング処理を行う．これは，座標に応じてマルチプレクサで適切な画素を選ぶことで実装できる．各層でパディングを行うため，モジュールの入出力で画像サイズは変わらず，ストリームが乱れることもない．

さらに，図 3.9 に示すように，座標に応じて水平及び垂直方向の反転を適用する．座標と反転種別の対応は表 3.1 に示した通りである．このような反転を各層で行うことで，ネットワークの前後で画像全体に対する反転を行う必要なく，3.3.2 項で述べた手法と同じ結果が得られる．両者の等価性を視覚化したものが，図 3.10 である．

反転後の小画像群は，ひとまとまりの画素群として，出力チャネルごとに用意された *forward* モジュール群の全てに与えられ，畳み込みとその他の必要な処理が行われた後，モジュール全体のレイテンシに基づいて調整された座標とともに出力される．これらの出力は，そのまま，次の層にあたる *layer* モジュールの入力となる．

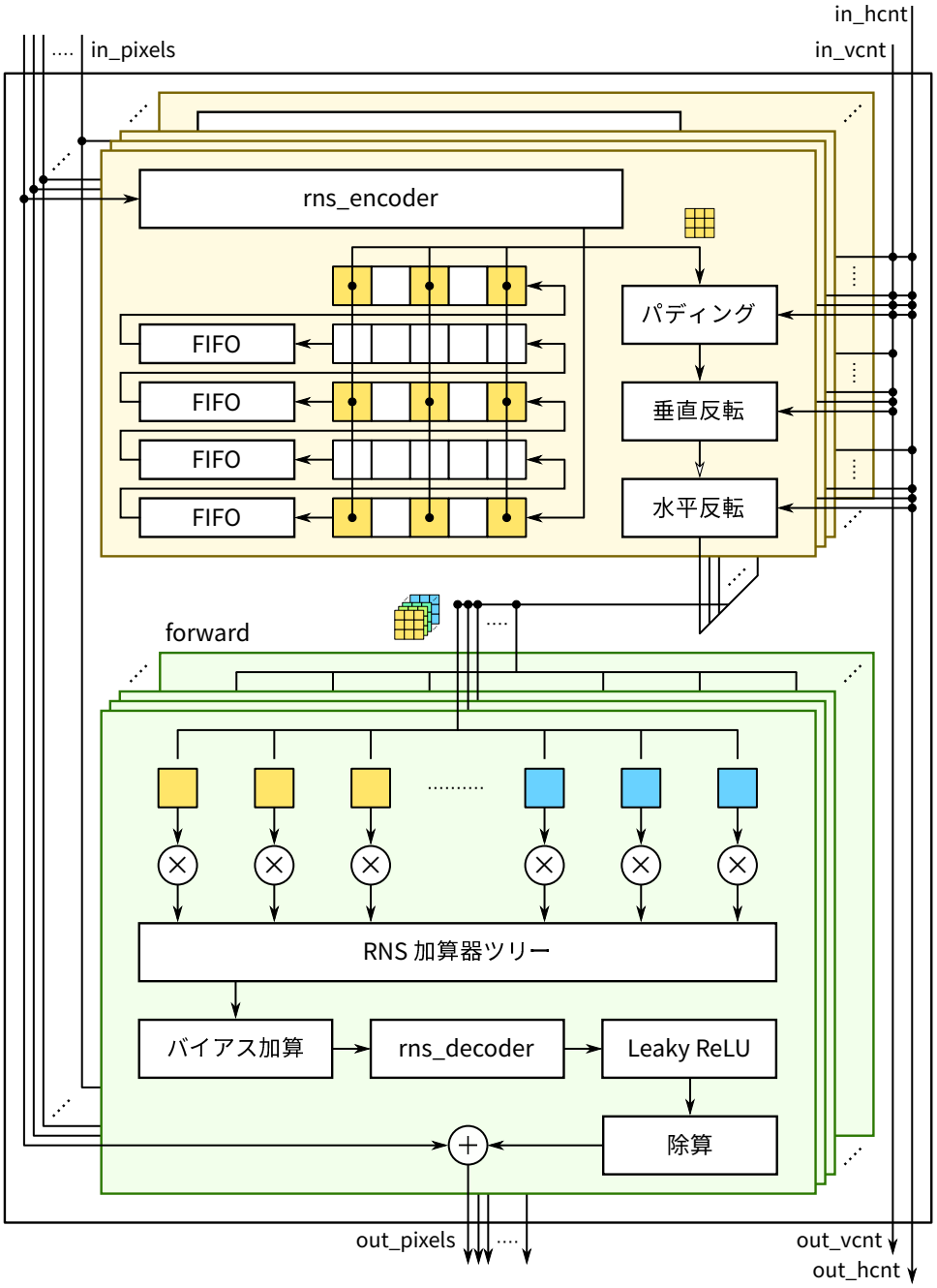


図 3.8: *layer* モジュールの構成図

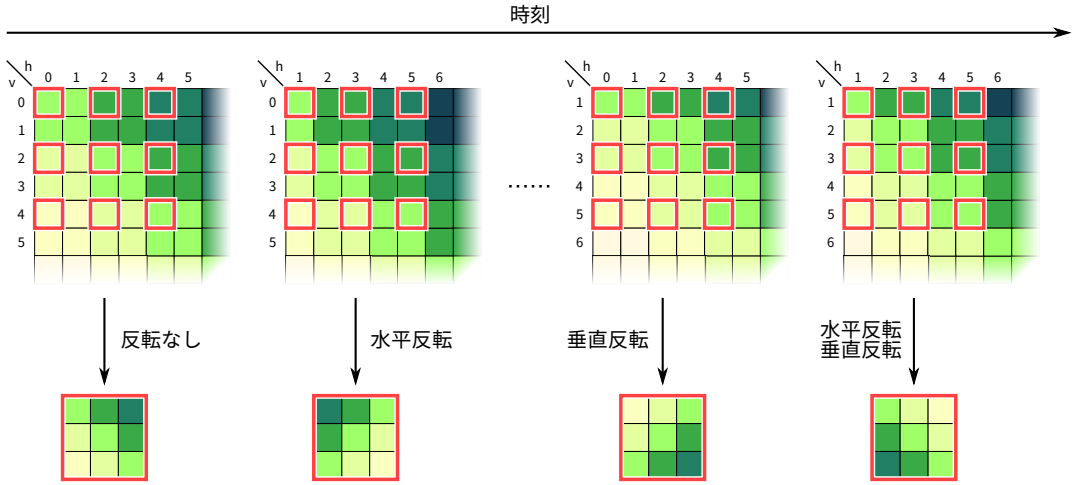


図 3.9: *layer* モジュールにおける反転

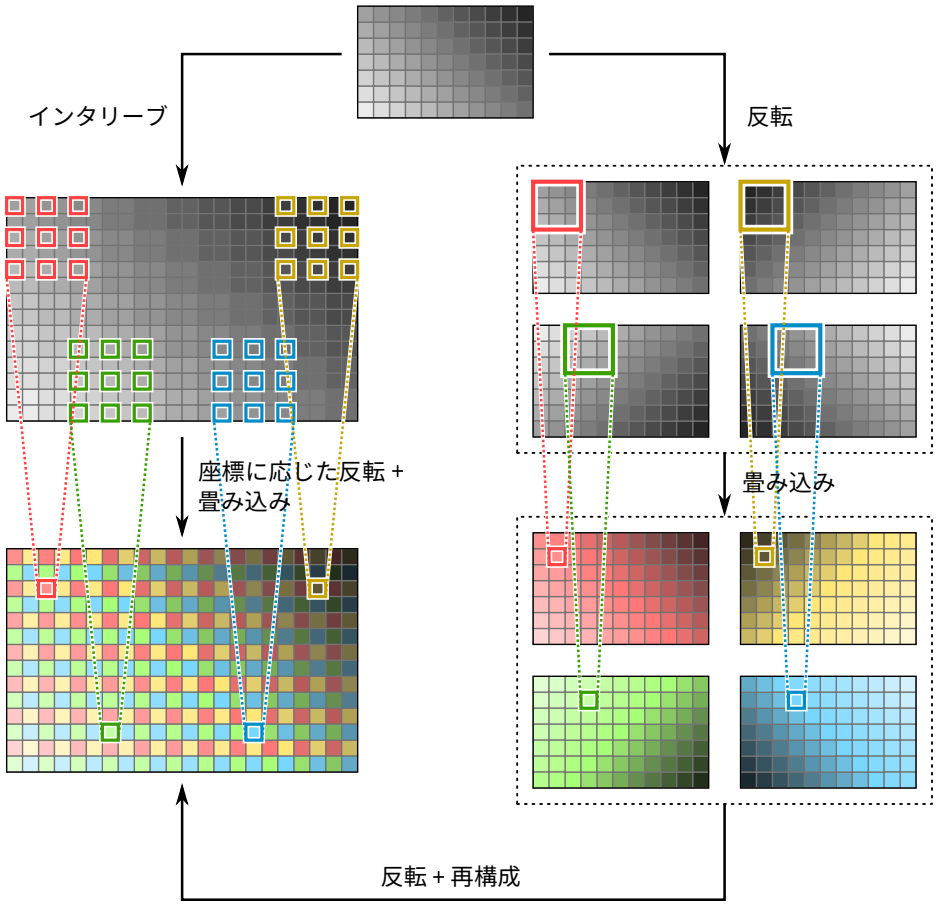


図 3.10: インタリーブされた各反転画像の流れ

3.4.5　*rns_encoder* モジュール

rns_encoder モジュールは，入力された 2 進数整数 $X = 2^n x$ を，RNS 表現 $\{x_i \mid x_i = X \bmod m_i\}$ に変換する機能を司る．基底 $M = \{m_i\}$ はパラメータにより指定できる．本システムが実装される Xilinx Virtex UltraScale FPGA では，LUT を 6 入力 1 出力 または 5 入力 2 出力 のいずれかで構成できることから，効率的な実装のため，法 m_i は 5 ビット以下のものに制限している．ここで，法 m が n ビットであるとは，任意の整数 x に対して $x \bmod m$ を n ビットの 2 進数で過不足なく表現できる，すなわち $2^{n-1}+1 \leq m \leq 2^n$ であることを意味するものとする．したがって，今回の実装では $2 \leq m_i \leq 32$ である．この制約のもとでも，法を $M = \{7,\ 11,\ 13,\ 17,\ 19,\ 23,\ 25,\ 27,\ 29,\ 31,\ 32\}$ とすれば，ダイナミックレンジ $D > 2^{47}$ となり，ニューラルネットワークの数値表現として実用上十分なダイナミックレンジが得られる．

本モジュールは，M に含まれる各々の法 m に対して，LUT を用いた剰余計算を行う．まず，入力値 X を下位から 6 ビットごとに区切り，分割された各ブロック $X_i\ (i = 0, 1, 2, ...)$ について，6 入力 LUT で剰余 $2^{6i}X_i \bmod m$ を求める．ただし，X は 2 の補数で表現された符号付き整数であるため，最上位ブロックの MSB は負の重みを持つ．続いて，モジュロ加算器ツリーを用いて全剰余の総和をとることにより，最終結果 $\left(\sum 2^{6i}X_i\right) \bmod m = X \bmod m$ を得る．X が 18 ビットである場合の構成を図 3.11 に示す．

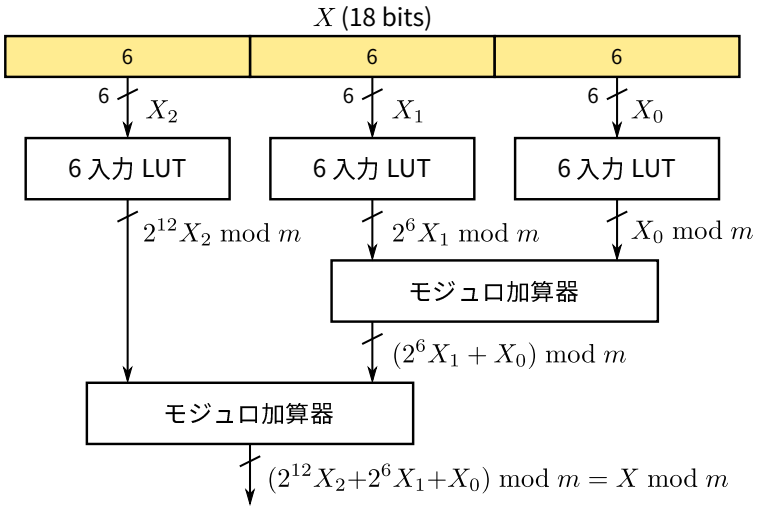
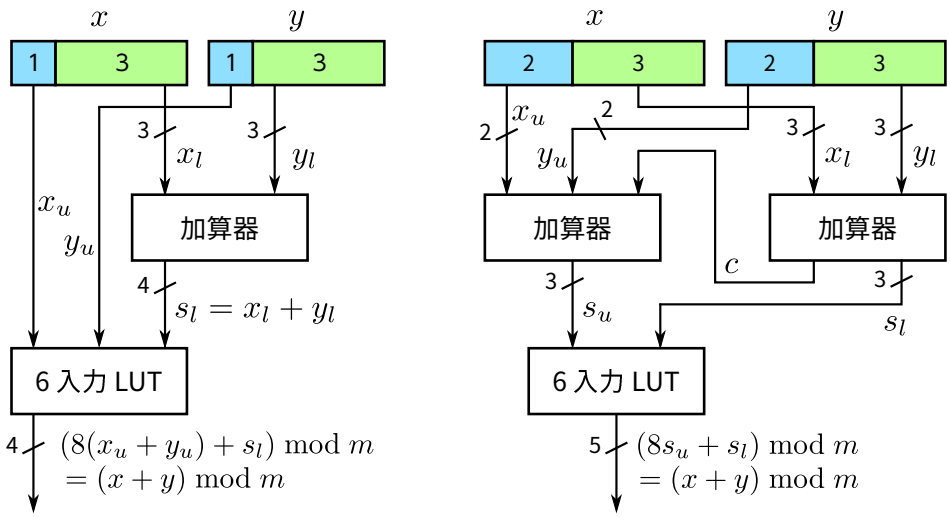


図 3.11: *rns_encoder* モジュールにおける剰余計算

3 ビットの法 $(5 \leq m \leq 8)$ に対するモジュロ加算器は，6 入力 LUT を用いることで容易に実現できる．出力が 3 ビットであるため，必要な LUT の数は 3 である．法が 4 ビット $(9 \leq m \leq 16)$ または 5 ビット $(17 \leq m \leq 32)$ である場合は，図 3.12 に示す通り，各入力 $x,\ y$ をそれぞれ，下位 3 ビット $x_l,\ y_l$ と残りの上位ビット $x_u,\ y_u$ に分割する実装となっている．

4 ビットの法においては，x_l と y_l を加算した後，得られた 4 ビットの和 $s_l = x_l + y_l$ と，残された各 1 ビットの x_u 及び y_u，計 6 ビットを 6 入力 LUT に与えて結果 $(8(x_u + y_u) + s_l) \bmod m = (x + y) \bmod m$ を得る (図 3.12 (a))．必要な LUT の数は 8 である．一方，

5 ビットの法においては，下位 3 ビット及び上位 2 ビットに分割した上で 2 数の加算を行い，得られた 6 ビットの和 $x+y$ を 6 入力 LUT に与えて剰余 $(x+y) \bmod m$ を得る (図 3.12 (b))．必要な LUT 数は 11 である．なお，実際に使われる LUT 数は，合成ツールの最適化により減少しうる点に注意されたい．例えば，法が 32 である場合，図 3.12 (b) に見られる剰余をとるための 6 入力 LUT は不要となる．



(a) 4 ビットの法　　(b) 5 ビットの法

図 3.12: モジュロ加算器の実装

3.4.6　*forward* モジュール

forward モジュールは，図 3.8 に示すように，層の出力値それぞれに対して割り当てられ，RNS 上での入力画素値群に対する畳み込み演算 (乗算と総和計算)，バイアス加算，2 進数整数への変換，活性化関数 (Leaky ReLU) の適用及びビット切り詰めを行う．なお，フィルタ係数及びバイアス値は， SystemVerilog の real 型 (64 ビット浮動小数点数) 配列で，パラメータとして与えられるよう設計している．

本モジュールではまず，入力された RNS 値と対応するフィルタ係数との乗算を行う．2.6 節で述べたように，これは法ごとのモジュロ乗算となるが，係数は固定値であるから，法が n ビットであるとき，この乗算は n 入力 LUT n 個を用いて容易に実現できる．論理合成時に，パラメータとして与えられた real 型係数 w は，小数部ビット幅に応じて整数に変換され，それに基づいて以下のような値を持つ LUT が自動的に構築される．

$$\mathrm{LUT}[x] = (x \times \lfloor 2^n w + 0.5 \rfloor) \bmod m_i$$

なお，$\lfloor x \times 2^n w + 0.5 \rfloor \bmod m_i$ とすることもできるが，固定小数点演算による実装と資源使用量を比較することを想定し，演算結果が同一になるようにしている．

続いて，RNS 加算器ツリーにより総和計算が行われる．RNS 加算器は，3.4.5 項にて説明したモジュロ加算器を，基底 M に応じて並列実装したものである．こうして求められ

た総和に，バイアスが加算される．バイアスは固定であるため，フィルタ係数との乗算と同様に，n 入力 LUT を用いて実装できる．ただし，3.3.4 項で述べたとおり，乗算によって小数部に相当するビット数が $2n$ に増加していることから，整数に変換する際の重みは 2^n ではなく 2^{2n} である点が異なる．

バイアスの加算後は，*rns_decoder* モジュールにより RNS 表現から 2 進数整数への変換が行われ，活性化関数 (Leaky ReLU) の適用と，小数部ビットの増加分を切り詰めるための 2^n による除算 (n ビット右シフト) が行われる．商は最も近い整数に丸められた後，ショートカット接続を介して層の入力値と加算され，出力される．なお，入力チャネル数と出力チャネル数が異なる場合，どちらか小さい方に合わせてショートカット接続される．つまり，前者の方が大きい場合，あふれた入力チャネルの値は接続されないし，後者の方が大きい場合，あふれた出力チャネルには何も加算されない．

3.4.7　*rns_decoder* モジュール

rns_decoder モジュールは，*rns_encoder* とは逆に，RNS 表現から 2 進数整数への変換を行う機能を備え，2.6 節にて示した変換式 (2.2) にあたる機能が，パイプラインで実装されている．図 3.13 がその構成図である．

本モジュールはまず，入力されたそれぞれの法における剰余 x_i と，対応する重み w_i との積のダイナミックレンジ D による剰余 $w_i x_i \bmod D$ を求める．法 m_i は高々 5 ビットであるから，この演算はテーブルを引くことにより容易に行える．

続いて加算器ツリーにより，それらの剰余の総和

$$s = \sum_{k=0}^{n-1} (w_k x_k \bmod D)$$

を求める．ただし，n は基底に含まれる法の総数である．この加算器ツリーは通常の整数加算器により構築されており，資源使用量の低減を目的として，一度に 3 値の加算を行う 3 項加算器ツリーとなっている．

最後に，s のダイナミックレンジ D による剰余を取ることで，2 進数整数 X を求める．

$$s \bmod D = \sum_{k=0}^{n-1} (w_k x_k \bmod D) \bmod D = \sum_{k=0}^{n-1} (w_k x_k) \bmod D = X$$

s と D はいずれも大きな整数であるので，3.4.5 項で述べたような手法での剰余計算は難しい．そこで本モジュールでは，$s \leq n(D-1) < nD$ であることに着目し，$\lfloor \log_2(n-1) \rfloor$ から 0 までの各整数 k を用いて，s が $2^k D$ 以上ならば $2^k D$ を引くという操作を順に行うことで，所望の結果を得る．この比較及び減算の回数は $\lfloor \log_2(n-1) \rfloor + 1$ で固定であるため，パイプラインの流れを乱すことはない．

なお，ここで得られた X は符号なし整数である．本来の符号付き整数に変換するためには，しきい値 t との比較を行い，$t < X$ ならば D を引くという操作を行う．

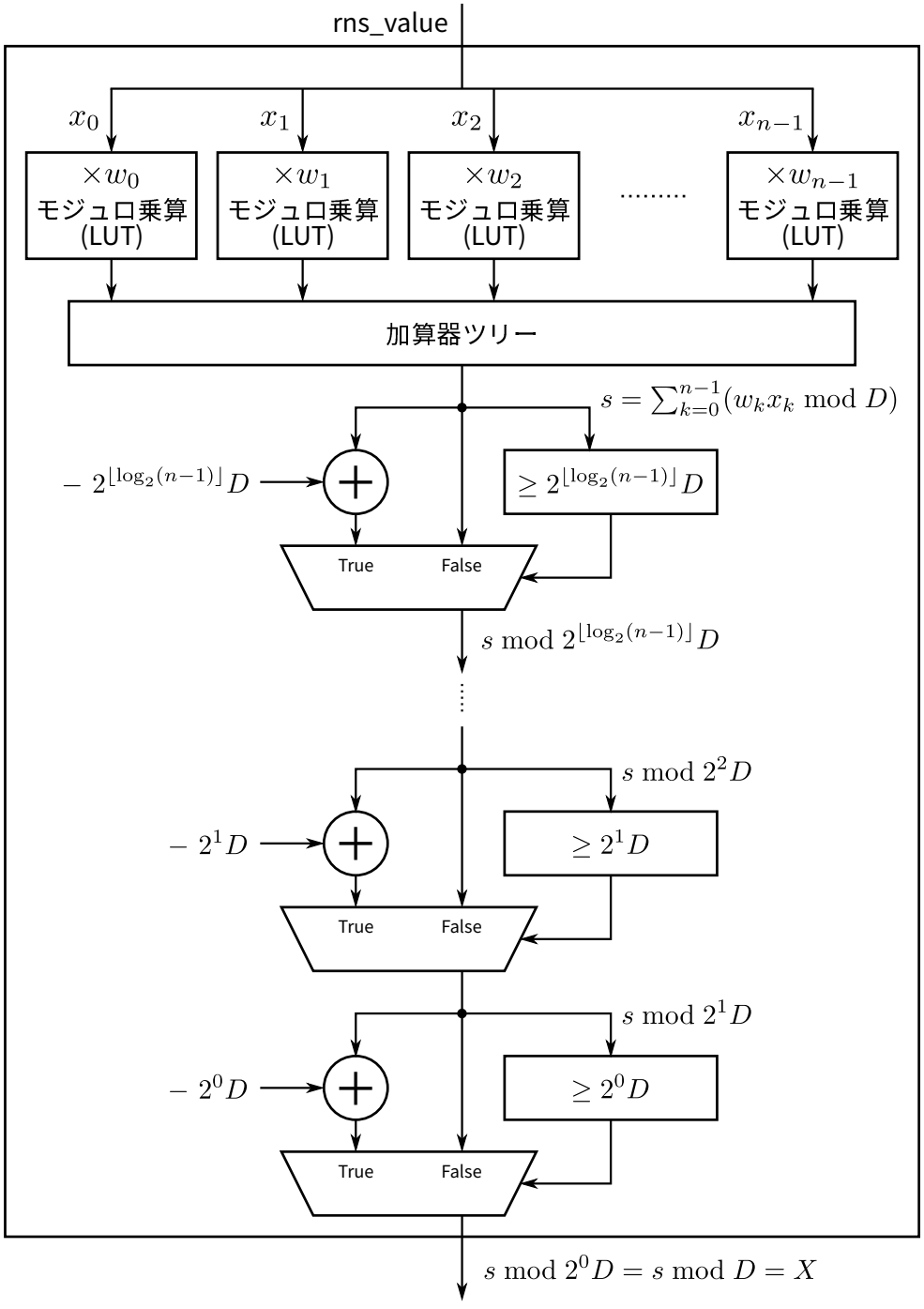


図 3.13: *rns_decoder* モジュールの構成図

3.5　学習

本節では，本研究の超解像システムに用いる畳み込みニューラルネットワークの学習手法について，ソフトウェアの実装面にも触れながら記述する．

3.5.1　ツール

学習には，Python 上で動作するニューラルネットワーク用のフレームワークである Chainer [39] 5.3.0 を用いる．Chainer では，NVIDIA 社の GPU 向けコンピューティングプラットフォーム CUDA を用いて，GPU により高速に学習を進めることができる．また，学習及び評価に必要となる各種の処理は，Python 上で数値計算ライブラリ NumPy 1.16.2，コンピュータビジョンライブラリ OpenCV-Python [40] 4.0.1.24，画像処理ライブラリ scikit-image [41] 0.12.3 を用いて行う．

3.5.2　データセットと学習手順

ネットワークの学習に用いる学習データセットと，超解像品質の評価に用いる評価データセットはそれぞれ，フル HD (1920×1080) の RGB カラー画像 300 枚及び 30 枚から成る．いくつかの画像例を図 3.14 に示す．また，データセットに含まれる画像のカテゴリの内訳は，表 3.2 に示すとおりである．

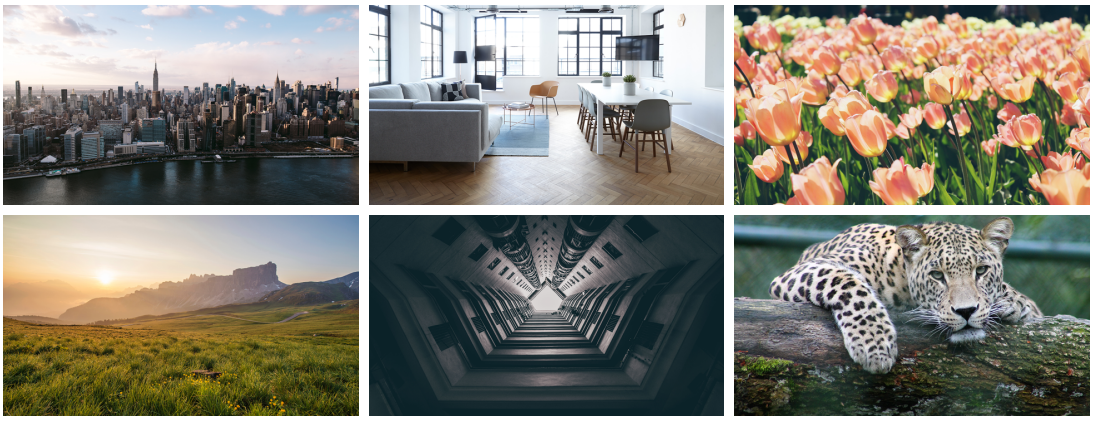


図 3.14: データセットに含まれる画像の例

学習は，Python で記述した学習プログラムを用いて行われ，データセット規模に対するスケーラビリティと処理効率を考慮して，ミニバッチ学習の形態をとる．なお，学習を高速化するため，学習データセット及び訓練データセットの全画像はあらかじめ全てメモリにロードしておく．

学習が始まると，Chainer で提供されている `chainer.iterators.SerialIterator` を用いて，学習データセットから画像が順次ランダムな順番で読み出される．これを基にして，学習に用いるミニバッチを生成する．ミニバッチの画像サイズは 256×256，ミニバッチサイズ (画像の枚数) は 8 としている．ミニバッチの生成手順を図 3.15 に示す．なお図

表 3.2: データセットに含まれる画像カテゴリの一覧

カテゴリ	画像数	
	学習	評価
街並み	60	6
インテリア	40	4
葉	40	4
花	40	4
風景	40	4
建築	40	4
動物	40	4
合計	300	30

中ではミニバッチを，ネットワークに与えるための訓練ミニバッチと，ネットワークの出力と比較するための教師ミニバッチに分けて表示している．

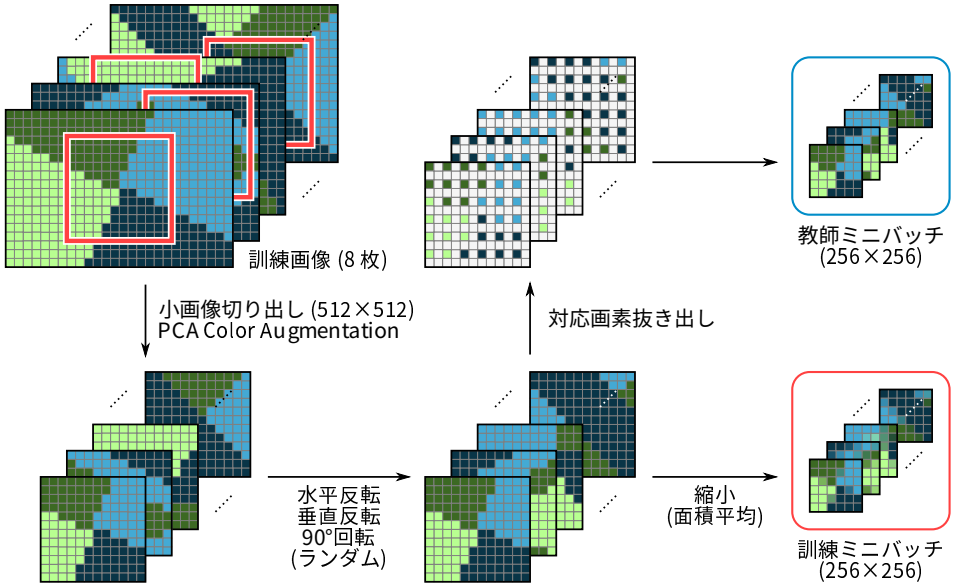


図 3.15: 反転手法におけるミニバッチの生成手順

まず，読み出された 8 枚の画像から，512×512 の小画像を切り出す．切り出し位置は，Python の `random.randint()` 関数を用いてランダムに決定する．次に，データ型を 8 ビットの符号なし整数から，Chainer で使用される 32 ビットの浮動小数点数 (`numpy.float32`) にキャストした後，データ拡張 (Data Augmentation) のため，2.5 節で述べた PCA Color Augmentation を適用する．ここで必要となる画像ごとの固有ベクトル $\boldsymbol{p}$ と固有値 λ は，学習開始時の学習データセットロード時に計算してメモリ上に保持しておく．続いて，さらなるデータ拡張のため，水平反転，垂直反転，90° 反転をランダムに組み合わせた合計 8

パターンの変形を加える．こうして生成された 8 枚の小画像を，OpenCV の `resize()` 関数により面積平均法 (`cv2.INTER_AREA`) で 256×256 に縮小した画像群が訓練ミニバッチとなり，垂直及び水平座標がいずれも偶数である画素 (3.3.2 項で述べた，4 画素のうち左上の画素に対応) のみを抜き出して生成した 256×256 の画像群が教師ミニバッチとなる．

以上の手順により生成されたミニバッチが Chainer に与えられ，一般的な誤差逆伝播法に基づき，ネットワークの学習が進められる．訓練ミニバッチをネットワークに与えて推論を行い，その結果と教師ミニバッチとの間の誤差が逆伝播される．誤差関数 (損失関数) には平均二乗誤差 (Mean Squared Error, MSE) を，学習率 (Learning Rate) の調整を行う最適化関数には Adam [42] を採用した．いずれも Chainer で標準関数として提供されているものである．Adam のパラメータについては，論文 [42] の推奨値かつデフォルト値でもある $\alpha = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$ をそのまま利用している．

3.5.3　比較用ネットワークの学習

品質比較のため，本研究の反転手法に基づくネットワークの他，3.2.2 項で述べた事前拡大手法，3.2.3 項で述べたサブピクセル再構成手法に基づくネットワークも同様の規模で構築し，学習を行う．基本的な学習手順はいずれも共通であるが，ミニバッチの生成手順のみ手法ごとに異なるため，本項で説明する．

事前拡大手法においては，図 3.16 に示すように，256×256 サイズでランダムに小画像を切り出し，データ拡張を適用したものがそのまま教師ミニバッチとなり，教師ミニバッチを一度面積平均法で 128×128 に縮小して，再びバイキュービック法 (`cv2.INTER_CUBIC`) により元のサイズまで拡大したものが訓練ミニバッチとなる．

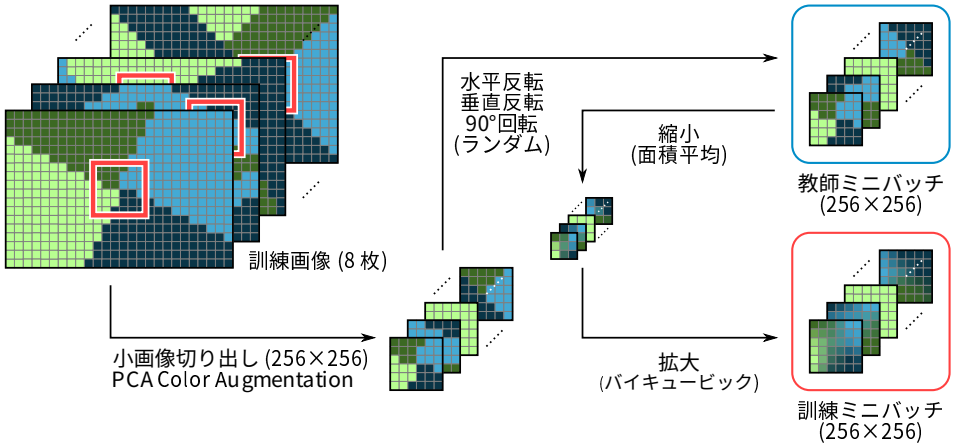


図 3.16: 事前拡大手法におけるミニバッチの生成手順

サブピクセル再構成手法では，図 3.17 に示すように，まずは反転手法と同じ方法で小画像の切り出しとデータ拡張を行う．これを面積平均法で縮小したものが訓練ミニバッチとなる点も同じである．一方の教師ミニバッチは，サイズを縦横半分にする代わりにチャネル数を 4 倍にする Space to Depth 変換により生成される．これは Chainer の `space2depth()` 関数を使い，スケーリングファクター引数 r に 2 を指定することで実現できる．関数

の入力画像を I とし，そのサイズを $(C,\ V,\ H)$ としたとき，出力画像 O のサイズは $(4C,\ V/2,\ H/2)$ となり，その対応関係は以下のように表される．

$$O(c,\ v,\ h) = I(c \bmod C, 2v + \lfloor c/(2C) \rfloor, 2h + \lfloor c/C \rfloor \bmod 2)$$

ただし，C は画像のチャネル数，V は高さ，H は幅である．今回の場合，$C = 3$ (RGB)，$V = H = 512$ となり，出力画像のサイズは $(12, 256, 256)$ となる．つまり，他の 2 手法と比べて，サブピクセル再構成手法では教師ミニバッチのチャネル数が 4 倍となっている．

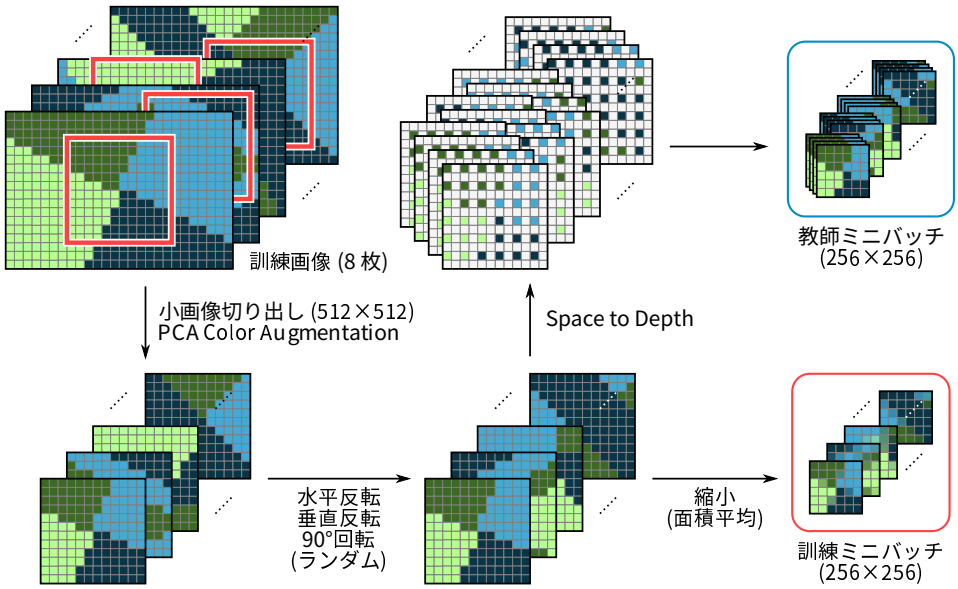


図 3.17: サブピクセル再構成手法におけるデータセットの生成手順

3.5.4　品質評価

1000 ミニバッチ (小画像 8000 枚相当) の学習が完了するごとに，評価データセットを用いた超解像品質の評価が行われる．まずは，学習用のミニバッチと同じ手順で入力画像と教師画像を生成し，入力画像をネットワークに与える．続いて，得られた浮動小数点型の結果を，最近値への丸めにより 8 ビット符号なし整数に変換してから，教師画像との誤差を基に，客観的画質評価指標を用いて品質を評価している．評価手法の詳細については 3.6 節に譲る．

評価の完了後は，評価値をファイルに保存する．また，評価値が過去最高である場合，または一定の評価間隔ごとに，処理結果の画像及びネットワークのパラメータも保存される．マルチプロセスで分散処理を行うことで，Python の Global Interpreter Lock (GIL) を回避し，評価に伴う学習処理の遅延を抑制している．プロセス間通信は，`multiprocessing` パッケージの `Queue` により実現する．

3.5.5　パラメータの変換

Chainer では，演算は単精度浮動小数点型で行われ，学習により得られるパラメータ (フィルタ及びバイアス) も浮動小数点型である．しかし，3.3.4 項で述べたように，実際のハードウェアでは RNS ベースの固定小数点演算が用いられる．そのため，学習中に保存されるパラメータを変換するためのプログラムを，Python で作成した．

3.4.6 項で言及したとおり，本システムのハードウェアモジュールは，real 型 (64 ビット浮動小数点数) のパラメータを受け入れ，論理合成時に自動で RNS 上の演算を実現する LUT を構築する．したがって，この変換プログラムが行うのは，SystemVerilog の `parameter` として認識されるように Chainer のパラメータを整形し，出力することのみである．具体的な処理の流れとしては，`numpy.ndarray` 形式の配列で保持されている Chainer のパラメータを，`numpy.ravel()` 関数で 1 次元に平坦化し，全要素を順次 `repr()` 関数を使って文字列に変換し，整形している．これにより生成されたパラメータファイルを，SystemVerilog のソースコード中で `include` することにより，パラメータの移行を容易に行える．

3.6　評価と考察

本節では，3.3 及び 3.4 節で述べた設計及び実装と，3.5 節で述べた学習手順に基づいて，他の手法との比較を交えながら，本研究の超解像システムの性能を評価する．

3.6.1　評価手法

本研究では，3.5.2 項で述べた評価データセットの入力画像をシステムに与え，得られた出力画像と教師画像との誤差を基に，超解像品質を評価する．評価指標には，ピーク信号対雑音比 (Peak Signal-To-Noise Ratio, PSNR) 及び Structural Similarity (SSIM) [43] を用いる．SSIM は構造的な類似性に着目した画質評価指標で，PSNR 等と比較して，主観画質評価との相関に優れているとされる．

画像 x 及び y の間の SSIM は以下の式により算出される．

$$\mathrm{SSIM}(x,\ y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)}$$

ここで，μ_x 及び σ_x^2 はそれぞれ x の平均及び分散であり，μ_y 及び σ_y^2 についても同様である．σ_{xy} は x と y の共分散であり，c_1 及び c_2 は値を安定させるためのパラメータである．SSIM は，2 画像が完全に同一である場合のみに最大値の 1 をとる．

評価データセットの各入力画像に対して超解像を適用し，scikit-image の `compare_psnr()` 及び `compare_ssim()` 関数を用いて PSNR 及び SSIM を算出する．各関数のパラメータには，`compare_ssim()` の `multichannel` を `True` とした点を除いて全てデフォルト値を利用している．この手順により求められた，評価データセットに含まれる 30 組のデータ全てにわたる PSNR 及び SSIM の平均値により，超解像の品質を評価する．なお参考として，OpenCV のバイキュービック補間により拡大した場合の平均品質は，PSNR が 32.2730，SSIM が 0.9221 である．

3.5.4 項で言及したように，Chainer による学習過程において，1000 ミニバッチごとに超解像品質の評価が行われる．このサイクルを 1 イタレーションと呼ぶことにする．本節ではまず，本研究の反転手法 (Flip) を用いたシステムに加え，関連研究において用いられている事前拡大手法 (PE) 及びサブピクセル再構成手法 (SP) のそれぞれについて，複数のネットワークサイズ (16/20/25) を組み合わせた 9 構成で 5000 イタレーションまで学習を進め，その過程で品質に生じる差異を評価する．その結果は 3.6.2 項で扱う．また，3.3.3 及び 3.5.2 項で言及したように，本研究では基本的に，ネットワークの活性化関数として Leaky ReLU を，学習時のデータ拡張として PCA Color Augmentation を適用するが，これらが及ぼす影響を検証するため，別途活性化関数として ReLU を用いた構成や，PCA Color Augmentation を無効化した構成も用意し，比較することとする．その結果は 3.6.3 項で扱う．

評価したシステム構成の一覧を表 3.3 に示す．ここで,「PCA CA」は PCA Color Augmentation の略である．また「フィルタパラメータ数」はフィルタ係数の総数を意味し，ネットワークの規模や演算量の目安となるものである．サブピクセル再構成手法のみ出力が RGB $\times 4 = 12$ チャネルあるため，他の 2 手法と比べ，同等の構成においてもパラメータ数がやや多くなっている．なお，実際のハードウェアにおいては，浮動小数点ではなく固定小数点で演算が行われるため，学習中に行われる評価とは品質に若干の差異が生じることに留意されたい．固定小数点演算を用いた場合の品質への影響は，3.6.4 項にて別途評価することとする．

表 3.3: 評価したシステム構成の一覧

構成	手法	出力チャネル数				PCA CA	活性化関数	フィルタパラメータ数
		1	2	3	4			
PE16+CA	事前拡大	16	16	16	3	Yes	Leaky ReLU	5472
PE20+CA		20	20	20	3			8280
PE25+CA		25	25	25	3			12600
PE25		25	25	25	3	No		12600
SP16+CA	サブピクセル再構成	16	16	16	12	Yes		6768
SP20+CA		20	20	20	12			9900
SP25+CA		25	25	25	12			14625
SP25		25	25	25	12	No		14625
Flip16+CA	反転	16	16	16	3	Yes		5472
Flip20+CA		20	20	20	3			8280
Flip25+CA		25	25	25	3			12600
Flip25		25	25	25	3	No		12600
Flip25+CA (ReLU)		25	25	25	3	Yes	ReLU	12600

3.6.2　超解像手法とネットワーク規模による品質の変化

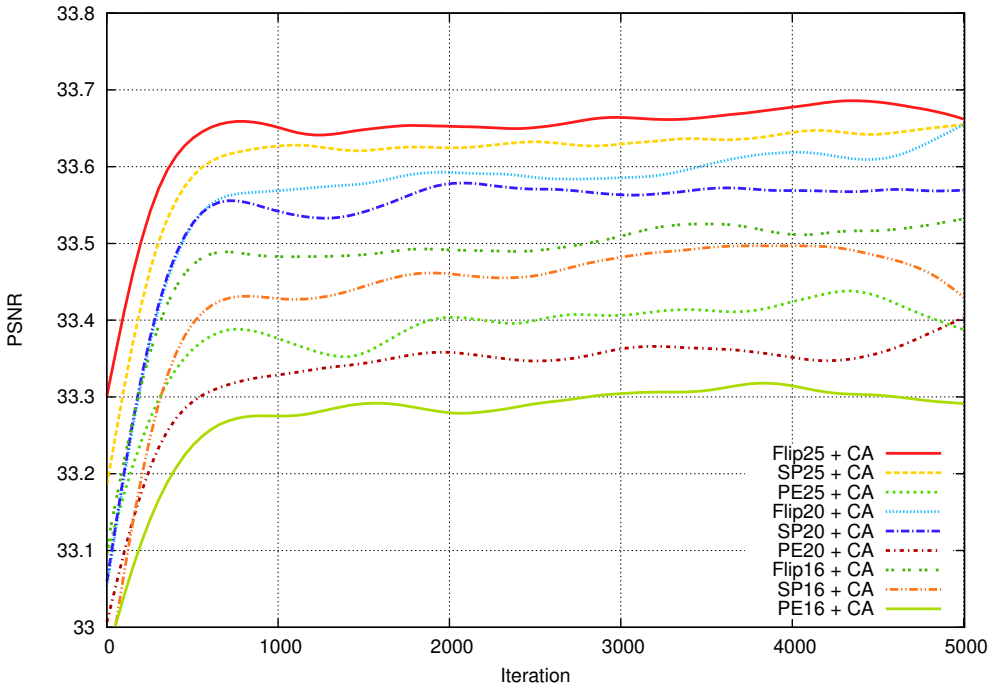
Flip+CA, SP+CA, PE+CA の 3 手法それぞれに対し，各層の出力チャネル数を 16, 20, 25 の 3 種類で変化させた計 9 構成について，学習の進行に伴う超解像品質の推移を図 3.18 に示す．ただし，生データは数値の変動が激しいため，視認性の観点から，図中の各グラフは近似曲線としている．近似曲線の生成は，グラフ描画ツールの gnuplot [44] で，`smooth acsplines` オプションを指定することにより行った．これは，3 次スプライン補間をベースにしたアルゴリズムで，重みと呼ばれるパラメータを大きくするほどデータ点を忠実に通る曲線を，小さくするほどデータ点から外れやすくなる代わりに滑らかな曲線を生成する．重みはデータ点ごとに指定できるが，今回の評価では一律で 10^{-9} を与えている．なお，この近似の影響により，一部のグラフで終端部に乱れが生じることがある点に留意されたい．

これらのグラフより，評価したいずれのシステム構成でも，バイキュービック補間 (PSNR: 32.2730，SSIM: 0.9221) を上回る品質を実現できており，同一手法では，ネットワーク規模が大きいほど品質が高いことが分かる．また，学習が進むにしたがって品質はさらに向上していくが，そのペースは次第に緩やかとなり，学習後半になるとほとんど変化がなくなることが見て取れる．

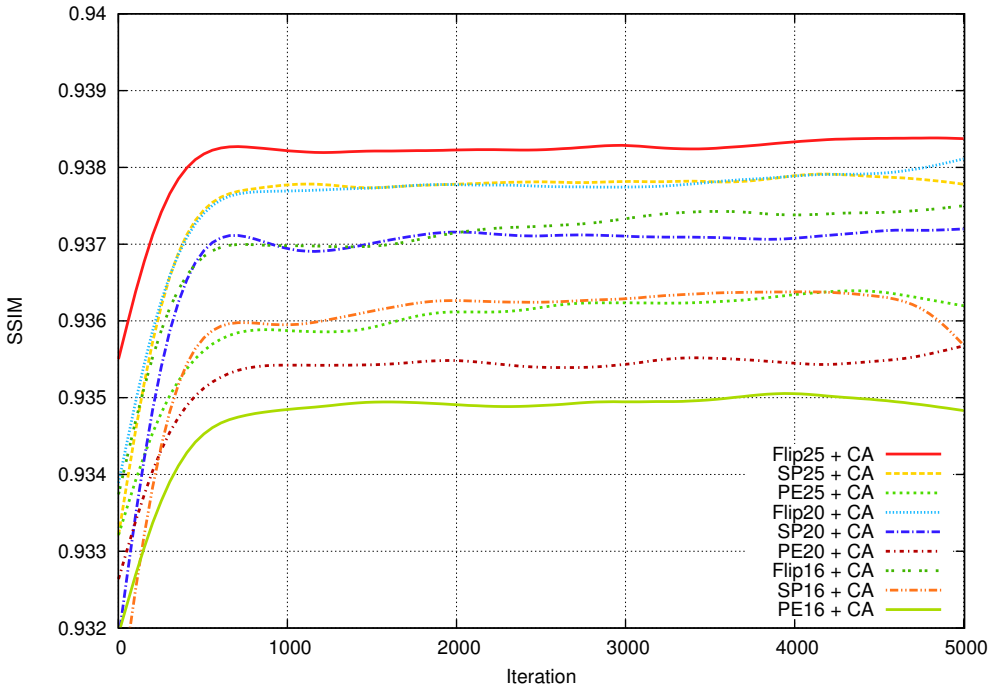
手法ごとに比較すると，同等のネットワーク規模である場合の品質は本研究の反転手法が最も高く，サブピクセル再構成手法，事前拡大手法の順に低下していく．全 9 構成中，Flip25+CA の品質が最も優れているが，Flip20+CA であっても，パラメータ数が約 1.8 倍である SP25+CA と SSIM において同等の品質を得られ．Flip16+CA については，パラメータ数が約 2.3 倍である PE25+CA をいずれの指標でも上回る品質を実現している．逆に，品質が最も劣る構成は PE16+CA である．事前拡大手法では，最大のネットワーク規模である PE25+CA であっても，他の手法の最小規模以下の品質しか得られていない．高解像度空間での処理を行う事前拡大手法では，3×3 のフィルタサイズは実質的に 1.5×1.5 相当と小さいため，十分な特徴抽出が困難となっている可能性が考えられる．

PSNR について，5000 イタレーションまでの学習過程における最大の値と，そのときのイタレーション数をまとめたものが表 3.4 である．図 3.18 のグラフと一致する結果が得られていることが分かる．また，最高の PSNR を達成したパラメータを使った実際の超解像画像から一部を切り出した例を，図 3.19 から図 3.21 に示す．いずれのシステム構成を用いても，バイキュービック補間より鮮明な画像を得られているが，各手法を比較すると，エッジの滑らかさや鮮鋭度において反転手法が最も優れており，サブピクセル再構成手法がそれに続く結果となっていることが分かる．一方事前拡大手法では，他の 2 手法と比べ，全体的に不鮮明な出力となっている．

なお，反転手法による構成において，エッジ周辺にまばらな薄いノイズが見られることがある．反転手法では，異なる反転の適用された 4 枚の入力画像を基に出力を得るため，入力画像によっては，隣接画素間での連続性が保たれない場合が生じることが原因であると考えられる．しかし，これらのノイズを含めた上で，これまで見てきたように PSNR や SSIM において反転手法の品質は他の手法と比較して高く，またネットワークの表現力が高まるにつれて，このノイズは減少していくことが期待される．



(a) PSNR



(b) SSIM

図 3.18: 基本 9 構成の超解像品質の比較

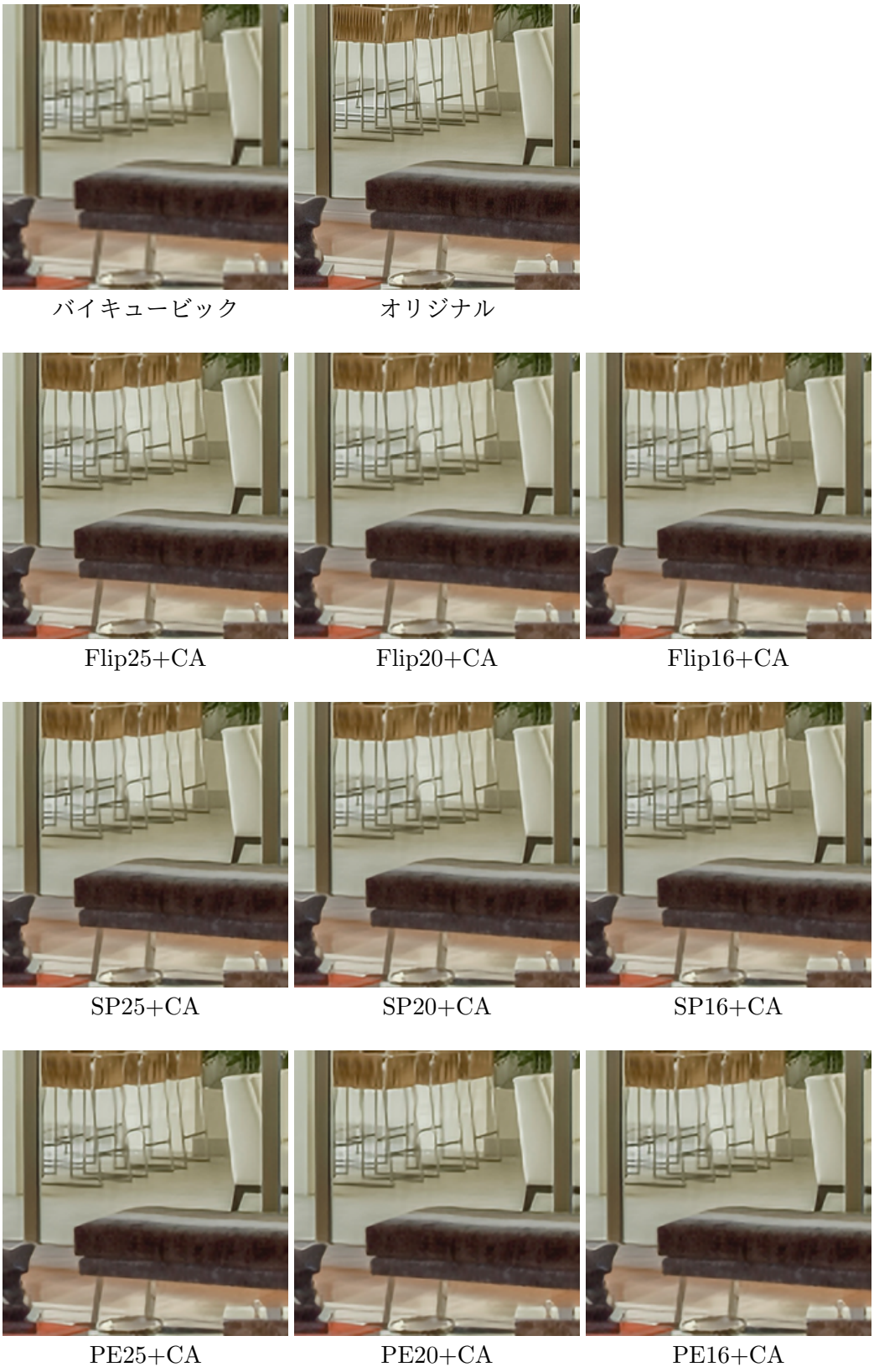


図 3.19: 基本 9 構成における超解像画像の例 (1)

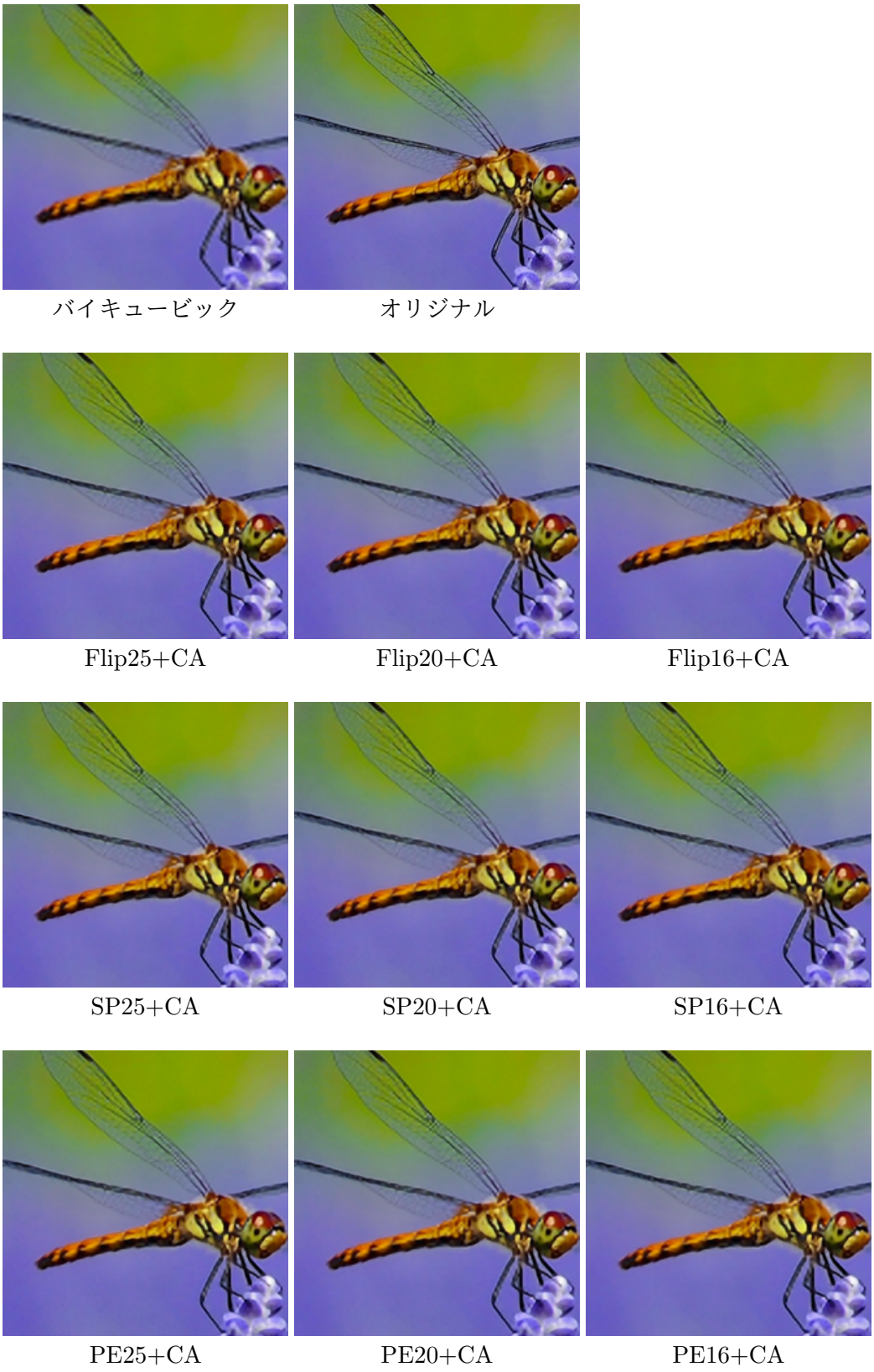


図 3.20: 基本 9 構成における超解像画像の例 (2)

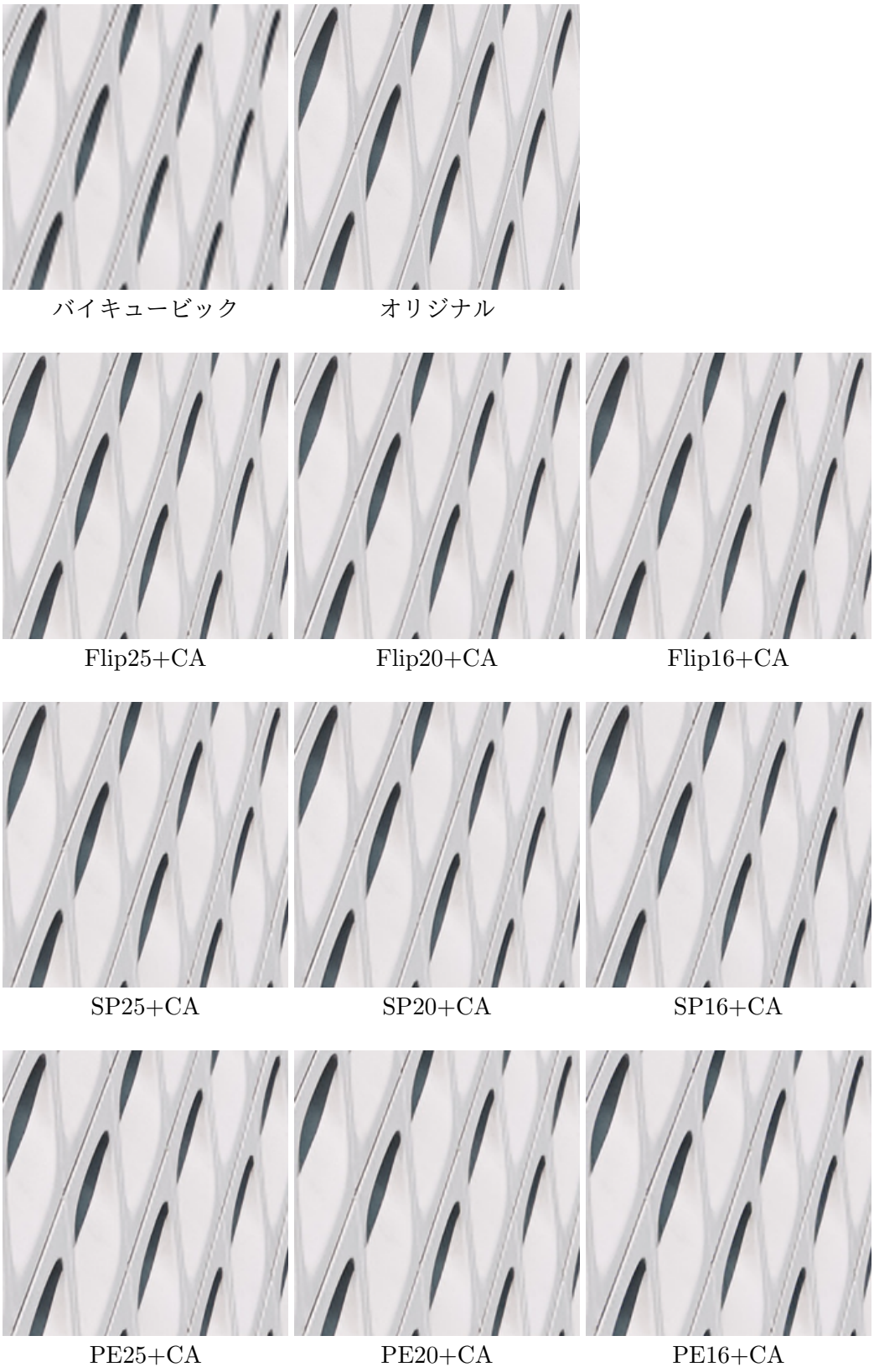


図 3.21: 基本 9 構成における超解像画像の例 (3)

表 3.4: 各構成の学習過程における最大の PSNR

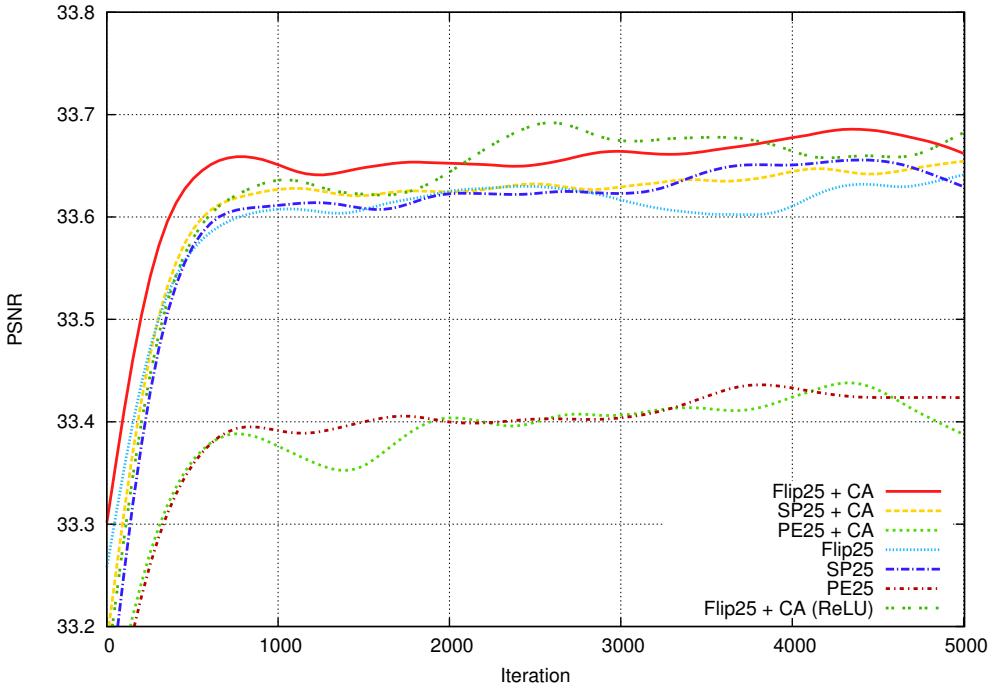
構成	PSNR	イタレーション
PE16+CA	33.4996	4109
PE20+CA	33.5893	4382
PE25+CA	33.6565	3935
PE25	33.6430	3713
SP16+CA	33.6408	4430
SP20+CA	33.7484	2234
SP25+CA	33.8375	4365
SP25	33.8351	4994
Flip16+CA	33.7020	3190
Flip20+CA	33.8107	4960
Flip25+CA	33.8864	3145
Flip25	33.8706	4772
Flip25+CA (ReLU)	33.8703	4863
Bicubic	32.2730	-

3.6.3　Leaky ReLU と PCA Color Augmentation の導入による影響

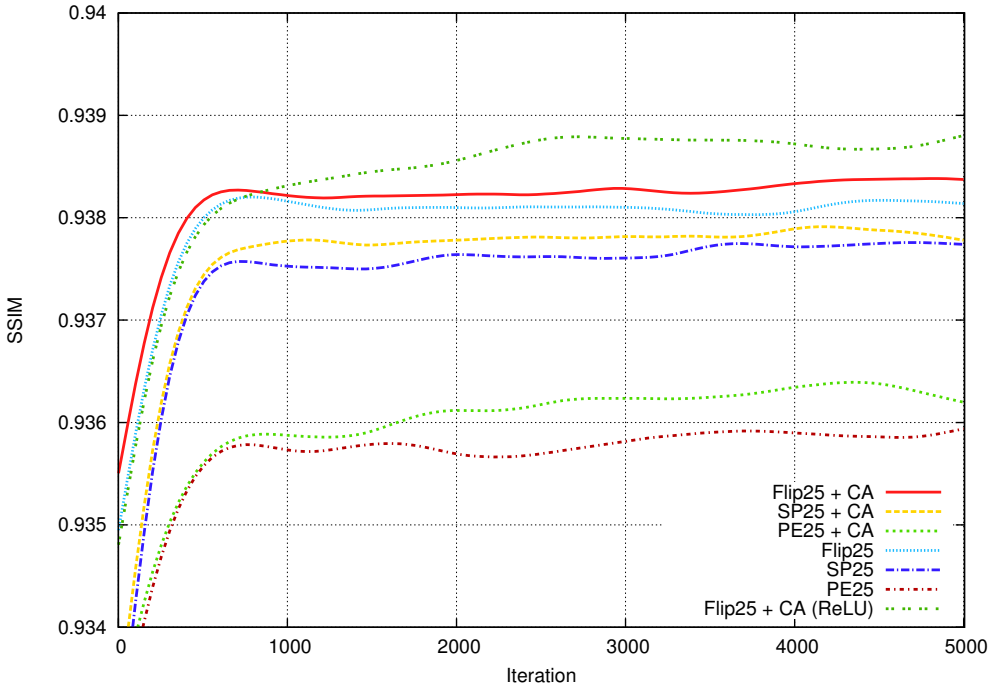
本システムでは，活性化関数として Leaky ReLU [21] $f(x) = \max(ax,\ x)\ \ (a = 0.25)$ を，データ拡張アルゴリズムの一つとして PCA Color Augmentation を採用している．そこで，これらの導入が及ぼす影響を検証するため，3.6.2 項で用いたシステム構成の一部について，活性化関数を一般的な ReLU $f(x) = \max(0,\ x)$ に置き換えたもの，あるいは PCA Color Augmentation を無効化したものを用意し，学習を進めることとした．その際の超解像品質の推移を示したものが，図 3.22 である．また，最高の PSNR を示した時点における超解像画像の例を図 3.23 にいくつか示す．

まず，Leaky ReLU を用いた基本構成である Flip25+CA と，活性化関数を ReLU に置き換えた Flip25+CA (ReLU) を図 3.22 で比較すると，SSIM では ReLU が上回る傾向を示したものの，PSNR ではほぼ同等となっている．これは，ピークの PSNR をまとめた表 3.4 からも確認できる．また，図 3.23 の超解像画像を見ると，ReLU ではエッジがよりシャープになる傾向が認められ，これが SSIM でやや上回る要因の一つであると考えられる．しかし，先述したエッジ周辺に発生するノイズは，Leaky ReLU の方が少ない．この差は，左側の椅子の画像で特に顕著である．

一方，各活性化関数で，ネットワークの各層における活性化関数直前の出力値分布を調べたところ，表 3.5 に示すように，ReLU では Leaky ReLU と比べて負の側に大きく偏っていることが明らかとなった．この現象は，3.3.3 節で述べたように，ReLU ではいかなる負の値も 0 としてしまうために起こると考えられる．例えば第 3 層を見ると，ReLU のダイナミックレンジは Leaky ReLU (Flip25+CA) の約 5.0 倍に達している．これはすなわち，RNS で必要なダイナミックレンジが約 5.0 倍になることを意味しており，資源使用量の増加につながる．



(a) PSNR



(b) SSIM

図 3.22: PCA Color Augmentation の有無と活性化関数の選択による超解像品質の比較

Flip25

Flip25+CA

Flip25+CA (ReLU)

図 3.23: PCA Color Augmentation と活性化関数の選択による超解像画像の比較

表 3.5: 活性化関数の直前における各層の値の分布

構成	層	最小値	最大値	分布幅
Flip25	1	-3.8580	3.6966	7.5547
	2	-13.7951	4.3780	18.1731
	3	-9.6116	4.5791	14.1907
	4	-2.6467	2.6428	5.2895
Flip25+CA	1	-3.8831	3.8661	7.7492
	2	-15.1395	4.6456	19.7851
	3	-12.6653	5.7528	18.4181
	4	-2.3151	1.9435	4.2587
Flip25+CA (ReLU)	1	-5.6665	2.1775	7.8440
	2	-50.9858	4.7033	55.6891
	3	-80.3211	11.2341	91.5553
	4	-7.2238	1.1327	8.3565

以上のことから，本研究の畳み込みニューラルネットワークでは，活性化関数として Leaky ReLU を用いることで，品質へ大きな悪影響を与えることなく，必要なダイナミックレンジを約 80% 低減でき，資源使用量を抑制できることが示された．

次に，PCA Color Augmentation について評価する．図 3.22 のグラフから，PCA Color Augmentation の導入により，品質が向上する傾向が見て取れる．特に SSIM においてその変化は一貫しており，Flip, SP, PE いずれの構成においても品質の向上が認められる．また PSNR についても，表 3.4 から，ピーク値は改善していることが分かる．本研究の超解像システムは，FPGA の資源制約を考慮して小規模なネットワークを採用しているが，それでも PCA Color Augmentation によるデータ拡張の導入は，超解像品質の向上に貢献することが確かめられた．

3.6.4　固定小数点演算による品質の変化

3.6.2 及び 3.6.3 項で行った評価は，浮動小数点演算によるものである．そこで本項では，実際のハードウェアで用いる RNS 上での固定小数点演算が，超解像品質に与える影響について評価する．なお，ここでの評価には，Python で記述したエミュレータを用いている．これは 64 ビット整数演算をベースに，RNS のダイナミックレンジに基づく値域制限を組み合わせてハードウェアの演算結果を厳密に再現するように設計されたプログラムであり，マルチプロセス処理と，NumPy が提供する高速な多次元配列演算により，ハードウェアのビヘイビアシミュレーションと比べて高速に動作し，超解像適用後の品質評価も容易に行える利点がある．

Flip25+CA (3145 イタレーション) 構成において，ネットワークの各層における入力値の小数部ビット幅を，8 から 12 まで変化させながら，超解像品質を評価した結果が表 3.6 である．ビット幅が大きくなるにつれて，概ね一貫して品質が向上していくことが分かる．また，実際の超解像画像の例を図 3.24 に示す．ここでは 8 ビットにおいても，視認でき

る差はほとんどない．ビット幅が 1 増えるごとに，必要なダイナミックレンジは 4 倍となるため，資源使用量を考慮すると，8 ビットの品質は許容可能であると考える．

表 3.6: 小数部ビット幅による品質の変化

小数部ビット幅	PSNR	SSIM
8	33.7193	0.9375
9	33.7071	0.9379
10	33.7522	0.9388
11	33.8506	0.9390
12	33.8482	0.9389
Float	33.8864	0.9397
Bicubic	32.2730	0.9221

3.6.5　ハードウェア実装

本項では，Flip16+CA，Flip20+CA，Flip25+CA の 3 構成を FPGA に実装し，資源使用量及び最大動作周波数を比較する．学習パラメータは，いずれの構成でも表 3.4 で示した時点のものを使い，小数部のビット幅には，3.6.4 項での評価結果を基に，8 ビットを用いることとする．RNS の構成パラメータ (法 M および正負のしきい値 t) は，表 3.7 に示すとおりであり，評価データセットに対する各層の出力値分布に基づき，オーバーフローが起こらないように設定している．基底中の法は互いに素であれば良いので，2 のべき乗の法を 1 つ組み込むことにより，3.4.5 項で述べたモジュロ加算器における剰余計算を不要にでき，資源使用量の削減につながる．

表 3.7: 各層における RNS の構成

層	Flip16+CA	Flip20+CA	Flip25+CA
1	$M = \{5, 7, 9, 11, 13, 32\}$ $t = 524288$	$M = \{5, 7, 9, 11, 13, 16\}$ $t = 327680$	$M = \{5, 7, 9, 11, 13, 16\}$ $t = 327680$
2	$M = \{5, 7, 11, 13, 17, 32\}$ $t = 786432$	$M = \{5, 7, 9, 11, 13, 32\}$ $t = 425984$	$M = \{5, 7, 9, 11, 13, 32\}$ $t = 360448$
3	$M = \{5, 7, 9, 11, 13, 32\}$ $t = 458752$	$M = \{5, 7, 9, 11, 13, 32\}$ $t = 458752$	$M = \{5, 7, 9, 11, 13, 32\}$ $t = 458752$
4	$M = \{5, 7, 8, 9, 11, 13\}$ $t = 172032$	$M = \{5, 7, 9, 11, 13, 16\}$ $t = 327680$	$M = \{5, 7, 8, 9, 11, 13\}$ $t = 163840$

また，RNS の導入に伴う資源使用量の削減効果を評価するため，固定小数点演算を用いるシステムも別途設計し，実装する．今回実装した RNS ベースのネットワークは，RNS 表現された値をビット列としてパッキングし，専用の演算モジュールを用いることで固定

8 ビット

10 ビット

12 ビット

Float

図 3.24: 小数部ビット幅による超解像画像の変化

小数点数と同じように扱える設計となっているため，演算モジュールを SystemVerilog の乗算記号 * や加算記号 + に置き換えることで，基本構造はそのままで固定小数点演算に切り替えられる．ただし，資源量を削減するため，各種加算器ツリーは 3 項加算器に置き換えて構築している．小数部は 8 ビットとし，演算結果を RNS と統一するため，乗算後のビット切り詰めは行わない．したがって，同じ構成であれば，レイテンシの違いを除いて固定小数点と RNS のいずれでも結果は完全に同一である．

シミュレーションには Cadence Xcelium 18.09 を，合成及び実装には Xilinx Vivado 2019.2 を用い，ターゲットとする FPGA は Virtex UltraScale xcvu095-ffva2104-2-e である．各構成における資源使用量及び最大動作周波数 $F_{\max}$ を表 3.8 にまとめる．なお，構成名の +CA は省略している．

表 3.8: 資源使用量及び最大動作周波数の比較

資源	固定小数点		RNS			使用可能
	Flip16	Flip20	Flip16	Flip20	Flip25	
LUT	210224 39.10%	303958 56.54%	236792 44.05%	320646 59.64%	456349 84.89%	537600
LUTRAM	1396 1.82%	1714 2.23%	2525 3.29%	3103 4.04%	3300 4.30%	76800
FF	245383 22.82%	355501 33.06%	231730 21.55%	310662 28.89%	430744 40.06%	1075200
BRAM	201 11.63%	249 14.41%	379 21.93%	445.5 25.78%	550.5 31.86%	1728
DSP	768 100.00%	768 100.00%	0 0.00%	0 0.00%	0 0.00%	768
CARRY8	27826 41.41%	43203 64.29%	1356 2.02%	1672 2.49%	2067 3.08%	67200
$F_{\max}$ (MHz)	149.2	142.2	155.7	148.2	146.7	-

表 3.8 に示されているように，固定小数点演算を用いた実装では，DSP ブロックの使用率が 100% となっている．これは，乗算演算子 (*) からツールが自動的に DSP を推論するためである．ただし，ネットワーク全体で必要な乗算回数は，Flip16+CA で 5472，Flip20+CA で 8280 であり，利用可能な DSP ブロック数である 768 を大きく超過しているため，`max_dsp` オプションで上限値を指定することで，超過分は論理ブロックで実現されるようにしている．また，加算が多用されているため，桁上がりを高速に伝播させるためのキャリーブロック (CARRY8) の使用率が DSP を除いて最も高く，Flip25+CA 構成については，論理合成後の使用率が 106.19% と搭載資源量を超過し，実装ができなかった．また，LUT の使用率も 94.20% と高値であった．

一方，RNS による実装の場合，演算が LUT ベースに置き換わるため DSP は使用されない．また，RNS 上の演算では桁上がりの伝播が必要ないため，CARRY8 は 2 進数整数への変換における減算等でのみ使用され，その量はわずかである．このため，固定小数点

演算では不可能であった Flip25+CA の実装も可能となっている．RNS 実装において使用率が最も高い資源は LUT であるが，演算の効率化により，一部を DSP にオフロードしている固定小数点実装と比較しても同程度の使用量に抑えられている．LUTRAM については増加が認められるが，これらは主に，移動ウインドウを実現するためのシフトレジスタとして使われている．また，BRAM についても 2 倍弱に増加しているが，これは 3.4.4 項の *layer* モジュールにおける FIFO が，RNS 変換後の画素値を扱うため，より大きなメモリサイズを必要とするためである．しかしいずれの使用率も，最大である LUT と比べて小さく，ネットワーク規模を制限するものではない．

システムの動作周波数については，いずれの構成においても入力部分で 33.3 MHz，以降のパイプラインで 133.3 MHz としている．この動作周波数でのフレームレートは約 60 fps となる．最大動作周波数については，表 3.8 にあるように，ネットワーク規模が大きくなるにつれて低下し，また同一構成では RNS 実装の方がより高くなっているが，実装に成功したいずれの構成においても動作周波数を上回っている．

レイテンシは，構成によりわずかな差異があるが，概ね 300 μs 程度である．このレイテンシは，大部分が各層でのバッファリングによるものであるため，ほぼ受容野のサイズに依存して決まり，ネットワーク規模の違い，あるいは固定小数点実装，RNS 実装の別にかかわらずほぼ一定である．このレイテンシは 1 フレームの長さ (約 16.7 ms) に対して十分に小さく，リアルタイム処理が実現できていると言える．

3.7　総括

本章では，畳み込みニューラルネットワークを用いたリアルタイムの動画像超解像システムの設計及び FPGA への実装を示し，その性能を検証した．本システムでは，既存研究に見られるような入力画像の事前拡大や，複数の出力チャネルを設けたサブピクセル再構成を行わず，代わりに水平及び垂直反転を組み合わせる反転手法により，ネットワークがより多くの情報を活用できるようにしている．また，ネットワーク内部の数値表現に Residue Number System (RNS) を用いることにより，加算及び乗算を小さな数のモジュロ演算に分解し，LUT での効率的な実装により資源使用量の削減を図っている．加えて，ネットワークの活性化関数に Leaky ReLU を採用することで，RNS に必要なダイナミックレンジを低減している．システムはパイプライン化され，低レイテンシでのストリーム処理を行うように設計されている．

畳み込みニューラルネットワークの学習は，ニューラルネットワークのフレームワークである Chainer を用いて行った．また比較対象として，既存研究の事前拡大手法及びサブピクセル再構成手法を含む，複数のシステム構成を用意し，それぞれ学習を進めた．その結果，本システムは PSNR 及び SSIM の両指標において，バイキュービック補間，ならびに他手法で同等のネットワーク構成を用いた場合と比べ，より高い品質を実現できることが示された．また，Leaky ReLU の導入効果を検討したところ，広く用いられている ReLU と比較して，超解像品質に大きな悪影響を与えることなく，必要なダイナミックレンジを最大約 80% 抑制できることが分かった．

続いて，実際のハードウェアにおける品質を，複数の小数部ビット幅で検証した．その結果，ビット幅が大きくなるにつれて品質は向上していくが，最小設定の 8 ビットであっ

ても，目視では品質の差がほとんど認識できないことが確認された．

最後に，固定小数点演算を用いる実装と，RNS ベースの実装を比較したところ，RNS 実装では CARRY8 ブロックの使用量が大幅に減少し，完全に同一の演算結果を保ちながら，固定小数点実装よりも大きな規模のネットワークを実現できたほか，最大動作周波数についても向上することが確認された．一方で LUTRAM 及び Block RAM の使用量については増加が認められたが，使用率は LUT と比べて小さく，ネットワーク規模を制限するものではなかった．加えて，本システムは 60 fps のフレームレートに対応でき，全体のレイテンシも約 300 μs と小さいことから，リアルタイムでの動画像超解像を実現できることが示された．

今後の展望として，より優れた品質を達成するために，実装の最適化やアーキテクチャのさらなる改良を目指す．具体例としては，演算精度やノード数等のパラメータを層ごとに最適化することや，Squeeze-and-Excitation Network (SENet) [45] 等の画像の大域情報を活用するアーキテクチャの導入が挙げられる．また，第 4 章で述べる手術画像セグメンテーションシステムのネットワークに導入した，畳み込み演算を空間方向とチャネル方向に分離する Depthwise Separable Convolution [4] の応用も，資源量の低減に寄与し，より大規模なネットワークの実現につながると考えられる．

第4章

手術画像セグメンテーションシステムの実装と評価

本章では，腹腔鏡手術における腹腔鏡の自動制御を目的とした，畳み込みニューラルネットワークによる手術画像セグメンテーションシステムの設計と GPU ベースの実装を示し，その性能を評価する．

4.1 背景と目的

腹腔鏡手術 (Laparoscopic Surgery) とは，腹腔鏡 (Laparoscope) と呼ばれる内視鏡器具を用いて，腹部に開けた 1 センチメートル前後の小さな穴から行う手術手技である．一般的な開腹手術と比べて侵襲性が低く，患者にとって痛みが少ない，出血量が少ない，回復期間が短いといった利点があることから，近年広く行われるようになっている．

腹腔鏡手術においては，腹腔鏡を保持し，適切な視野が得られるように位置や向きを調整するスコピストと呼ばれる技師が，執刀する外科医とは別に必要となる．一方，近年医療分野においては手術支援ロボットの活用が急速に広がっている．そこで，スコピストの役割をロボットにより代替できれば，スコピストの同伴を要せず施術が可能となる．これは，医師の地域偏在が問題となる中，離島医療の現場等，十分な医師の確保が難しい状況において有益であると考えられる．

このような状況を踏まえ，長崎大学工学部，長崎大学病院，中央大学理工学部が共同で，腹腔鏡の自律的制御を可能にする腹腔鏡手術支援システムの開発を進めることとなった．腹腔鏡制御の自動化にあたっては，中央大学で開発が進められている制御用のロボットに加えて，入力画像に基づいて適切なカメラアングルを決定するシステムが必要である．これを実現するためのアルゴリズムには様々なものが考えられるが，そのうちの一つとして，各画素を意味に基づいて分類するセマンティックセグメンテーション (Semantic Segmentation) を用い，画像中の各臓器の位置や距離を認識するという手法が挙げられる．

そこで本研究では，腹腔鏡手術として広く行われている，腹腔鏡下胆嚢摘出術を対象とした画像セグメンテーションシステムを設計し，その性能を評価する．このシステムは畳み込みニューラルネットワークを用いて実現されており，その学習にあたっては，長崎大学の協力を得て，外科医の手によりアノテーションされた学習データセットを利用している．しかしながら，データセットの作成は高度な専門性を要する作業であるため，現時点でのデータセットは小規模なものに留まっている．これは 2.3 節で述べた過学習を引き起

こし，システムが実際の現場で正しく機能するのを妨げてしまう．このため本研究では，工夫されたネットワーク構造を導入することで，過学習問題の軽減を図る．

4.2　関連研究

畳み込みニューラルネットワークを用いたセマンティックセグメンテーションのアーキテクチャは，数多く提案されている．本節では，それらの関連研究について，代表的なものを中心に簡単にまとめる．また，大きな畳み込みニューラルネットワークにおいて問題となる，高い演算負荷や膨大なパラメータサイズを軽減するために提案された，各種の手法についても併せて紹介する．

4.2.1　Fully Convolutional Network

J. Long らによって提案された Fully Convolutional Network (FCN) [46] は，ネットワーク中に全結合層 (Fully-Connected Layer) を含まず，全ての層が畳み込み層により構成されている点を特徴とするセマンティックセグメンテーションネットワークである．

画像分類等を目的とする従来のネットワークモデルでは，プーリング層を用いて大局情報を小さなサイズの特徴マップに集約していき，最終段にある全結合層により，クラスごとの尤度を推論していた．この手法では空間的情報が失われるため，画素単位での分類は行えない．そこで FCN では，全結合層を 1×1 の畳み込み (Pointwise Convolution) に置き換えることで，空間情報を保持した尤度マップを得るようにしている．ただし，プーリング層を経たことで解像度が低下しているため，このままでは粗い尤度マップしか得られない．そこで，スキップ接続 (Skip Connection) により，ネットワークの前段における解像度の高い特徴マップを参照できるようにし，これと粗い尤度マップを組み合わせてアップサンプリングを行うことで，画素単位の尤度マップが得られる仕組みとなっている．

4.2.2　SegNet

V. Badrinarayanan らが提案した SegNet [47] は，大局情報をプーリングにより低解像度空間に集約するエンコーダと，失われた画素単位の情報をアップサンプリングにより復元するためのデコーダを対称な形で組み合わせた，エンコーダ・デコーダ (Encoder-Decoder) 型のセマンティックセグメンテーションネットワークである．

FCN のスキップ接続と異なり，エンコーダで用いられている最大値プーリングで選ばれた最大画素の位置 (Pooling Indices) を保存しておき，デコーダでアップサンプリングする際に参照して，対応する位置に値を受け渡すことで，画素単位の位置情報を保持できるように設計されている．Pooling Indices は特徴マップそのものよりもデータサイズが小さいため，スキップ接続と比べて省メモリでの処理が可能である．

4.2.3　U-Net

U-Net は，生物医学領域の画像セグメンテーションを対象として，O. Ronneberger らが発表したセマンティックセグメンテーションネットワークである [48]．SegNet と同様のエンコーダ・デコーダ型構造をとりつつ，FCN のスキップ接続をより積極的に取り入れ，様々なスケールの特徴を効果的に利用できるようになっている．

図 4.1 に U-Net の構成図を示す．デコーダ部分は，転置畳み込み (Transposed Convolution) を用いたアップサンプリング畳み込みにより拡大された特徴マップに，エンコーダ部分からスキップ接続により渡された同サイズの特徴マップを連結した上で，さらに畳み込みを行い統合することを繰り返しながら，最終的に各クラスの尤度マップを得る仕組みとなっている．ただし，畳み込みの際にパディングがなされないため，そのサイズは入力画像よりやや小さくなる点に注意されたい．この仕組みにより，スケールの異なる特徴を融合させた処理が可能となる．

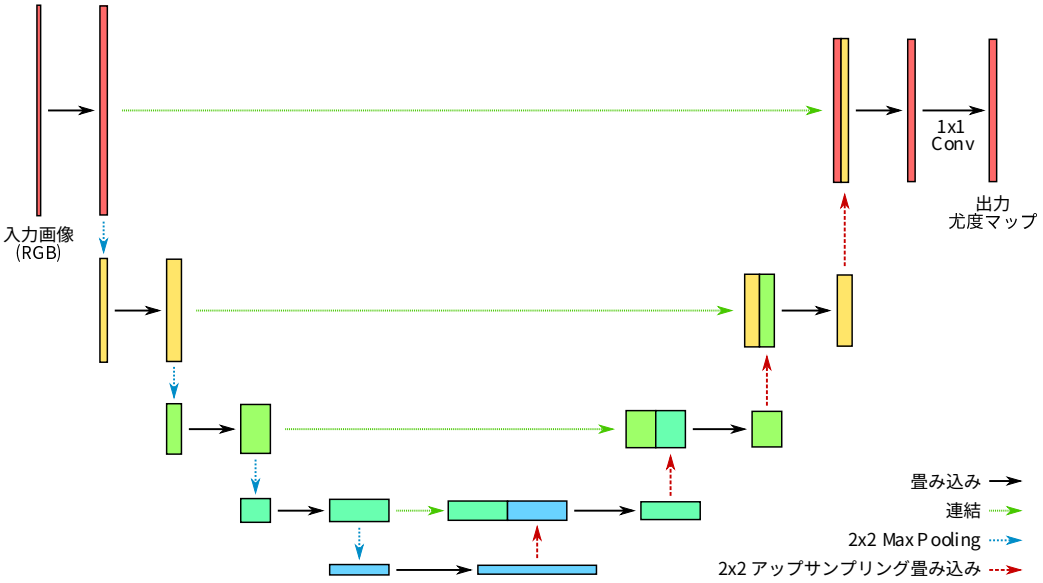


図 4.1: U-Net の構成図

4.2.4　PSPNet

H. Zhao らは，コンテキストの集約にピラミッドプーリングモジュールと呼ばれる構造を導入した，PSPNet と呼ばれるセグメンテーションシステムを発表した [49]．

PSPNet では，まず，ResNet [23] に Dilated Convolution [50] を組み合わせて構築した，エンコーダに相当するネットワークにより，特徴マップを抽出する．なお Dilated Convolution は，フィルタの画素間に間隔を空けて行う畳み込みであり，間隔を広げることで，プーリングのように解像度を落とすことなく，広い範囲の特徴を取り込みやすくする．続いてこの特徴マップは，ピラミッドプーリングモジュールに与えられる．ここでは，複数のスケールでのプーリングを適用して，それぞれの結果を畳み込み層に通し，アップ

スケーリングにより元のサイズに戻した上で連結するという処理を行う．これにより得られた特徴マップに対してさらに畳み込みを行うことで，大域的なコンテキストと局所的なディテールを集約したセグメンテーションを可能にしている．

4.2.5　ネットワークの軽量化を実現するアーキテクチャ

ネットワークの構造そのものの改善に加えて，畳み込みの仕組みに工夫を加えることで，大規模なネットワークにおいて問題となる，膨大な計算量やパラメータサイズの削減を図る研究が多く行われている．例えば，Xception [3] や MobileNet [4] では，Depthwise Separable Convolution と呼ばれるテクニックを導入し，軽量なネットワークを実現している．これは，従来の畳み込み層を Depthwise Convolution (チャネルごとに独立した畳み込み) と Pointwise Convolution (1×1 フィルタを使った畳み込み) に分離し，効率的な特徴抽出を実現するものである．

Depthwise Separable Convolution では，Pointwise Convolution の部分が計算量やパラメータ数で支配的になりやすい．そこで，Pointwise Convolution の部分を，Grouped Convolution などの手法により，小さな Pointwise Convolution の集まりに分解して近似することでさらなる軽量化を図った，ShuffleNet [6] や MobileNetV2 [5] といった改良アーキテクチャも提案されている．

4.3　設計

本節では，本研究における，過学習の軽減に配慮した手術画像セグメンテーション用畳み込みニューラルネットワークの設計について述べる．

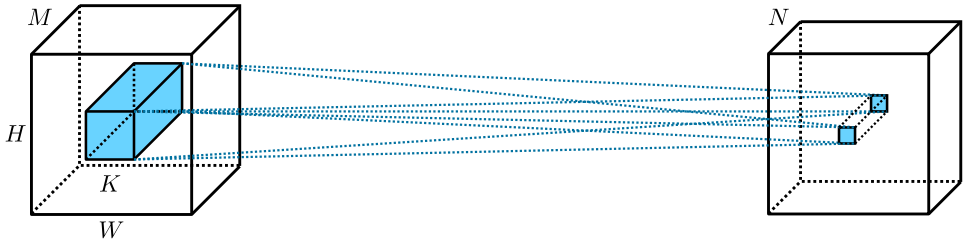
4.3.1　設計指針

4.2 節で言及したものを含め，既存のセグメンテーションシステムの多くは大規模な学習データセットに依存している．しかしながら，このようなデータセットを用意するには多くのコストと手間がかかるため，必ずしも常に利用できるわけではない．不十分なデータセットは，ネットワークの過学習を引き起こし，実環境下におけるシステムの有用性を損ねる．そこで本研究では，過学習を軽減し品質を向上させるネットワーク構造の工夫を導入し，その性能を評価することとする．

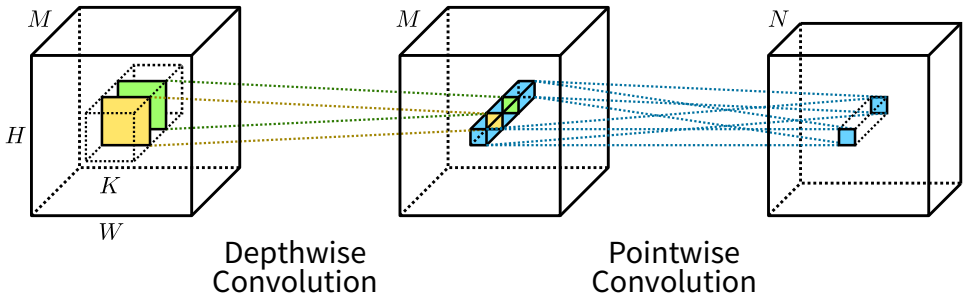
4.3.2　Depthwise Separable Convolution

Depthwise Separable Convolution (以下単に Separable Convolution，あるいは SepConv と呼ぶ) は，図 4.2 に示すように，通常の畳み込み層を，深さごとの畳み込み (Depthwise Convolution) と点ごとの畳み込み (Pointwise Convolution) に分離するテクニックである．これは，空間方向 (同一チャネルにおける近接画素間) の相関と，深さ方向 (チャネル間) の相関は独立しており，互いに分離可能であるという仮定に基づいている．[3] や [4] にて示されているように，Separable Convolution の導入により，推論精度に大きな低

下を引き起こすことなく，畳み込みニューラルネットワークの演算量やパラメータサイズを大幅に削減できる．



(a) 通常の畳み込み



(b) Separable convolution

図 4.2: 通常の畳み込みと Separable Convolution の比較

フィルタサイズが $K \times K$ であり，入力と出力のチャネル数がそれぞれ M と N であると仮定すると，通常の畳み込み層 (図 4.2 (a)) におけるフィルタパラメータの総数は K^2MN となる．一方 Separable Convolution では，Depthwise Convolution において K^2M，Pointwise Convolution において MN，合わせて $K^2M + MN = M(K^2 + N)$ までパラメータ数が削減される．フィルタサイズは $K = 3$ が用いられることが多いため，一般的なネットワークでは K^2 と比べて N がはるかに大きく，このため，Pointwise Convolution がパラメータ数において支配的になりやすい．

[5] や [6] では，このような場合に，Pointwise Convolution の分解が有効であることが示されている．例えば [6] においては，Grouped Convolution とチャネルのシャッフル操作を組み合わせている．Pointwise Convolution を G 個のグループに分割して行うことで，パラメータ数は $(M/G) \times (N/G) \times G = MN/G$ まで軽減される．その後，シャッフル操作によりグループ間の特徴を混ぜ合わせることで，後段の層で異なるグループ間の関係性を利用できるようにしている．しかし，本研究における設計では，学習データセットの規模が小さいことから，各層のチャネル数は 20 程度と比較的小さく設定している．したがって，Pointwise Convolution の分割は相対的に効果が小さいと考えられるため，シンプルな Separable Convolution のみを採用する．

Pointwise Convolution に引き続き，Leaky ReLU [21] $f(x) = \max(ax,\ x)\ \ (0 < a < 1)$ が活性化関数として適用される．今回，$a = 0.25$ としている．一般的な ReLU $f(x) = \max(0,\ x)$ と比べ，負の領域でも 0 でない傾き a を持つため，深いネットワークでも

勾配消失を起こしにくいという利点がある．なお，Depthwise Convolution と Pointwise Convolution の間では活性化関数は適用されない．

4.3.3　ネットワーク構造

ネットワークの基本構造としては，[47] や [48] 等で見られるエンコーダ・デコーダモデルを採用する．図 4.3 に示すように，ネットワーク全体は，*Extract*，*Reduce*，*Merge* と名づけた 3 つのサブネットワークと，1×1 の畳み込み層 (Pointwise Convolution) から構成されている．ただし，図中の (S, C) という表記法は，特徴マップのサイズが $S \times S$，チャネル数が C であることを示している．

このネットワークでは，小規模な学習データセットにより引き起こされる過学習の問題を軽減するため，再帰的な構造を取り入れている．これは，サブネットワークのうちの 2 つである *Reduce* と *Merge* を繰り返し再利用するものである．それぞれの利用回数を「再帰レベル」と呼ぶこととし，L $(= 1,\ 2,\ ...)$ で表す．なお，L はパラメータにより任意に変更できる．図 4.3 における再帰レベルは $L = 3$ である．

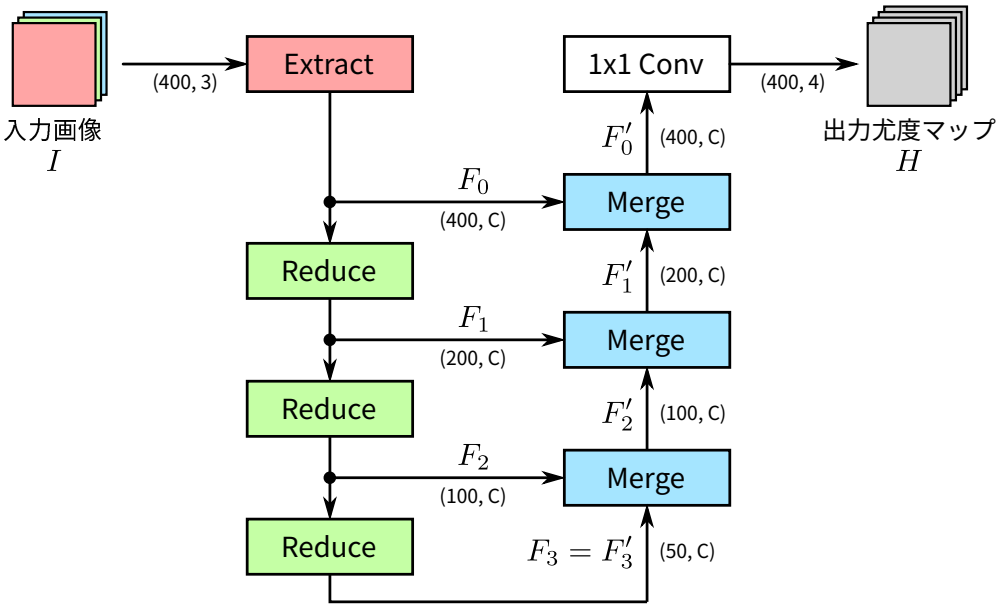


図 4.3: 再帰レベル $L = 3$ の場合のネットワーク構成図

まず，入力された RGB 画像は *Extract* サブネットワークを通り，C チャネルの特徴マップに変換される．なお，図 4.3 では入力画像のサイズが 400×400 となっているが，これは 4.4 節にて述べる学習時に用いられるサイズである．ネットワークには全結合層が含まれないため，$S = 2^L k$ $(k \in \mathbb{N})$ を満たすいかなるサイズの画像も受け入れることができる．

続く *Reduce* サブネットワークは，$2S \times 2S$ サイズの特徴マップを $S \times S$ の特徴マップに集約する役割を担う．*Reduce* は計 L 回適用され，入力画像の大局的なコンテキストを考慮しやすくする．ここで，F_n $(0 \leq n \leq L)$ を n 個目の *Reduce* が出力した特徴マップとする．ただし，F_0 は *Extract* の出力とする．すると，入力画像を I としたとき，F_n は

以下のように定式化できる．

$$F_0 = \mathrm{Extract}(I)$$
$$F_{i+1} = \mathrm{Reduce}(F_i) \quad (0 \leq i \leq L-1)$$

なお，いずれの F_n も出力チャネル数は C である．

残る *Merge* サブネットワークでは，まず F_L と F_{L-1} を入力とする．前者は内部でアップサンプリングされた後，後者と連結されて $2C$ チャネルの特徴マップとなり，続く畳み込み層により C チャネルの出力特徴マップに統合される．この出力は，同様の仕組みにより今度は F_{L-2} と統合される．このようにして *Merge* を L 回適用することにより，*Extract* と *Reduce* が出力した各スケールの特徴マップ群 $(F_0,\ F_1,\ ...,\ F_L)$ は単一の特徴マップに統合される．ここで，n 個目の *Merge* の出力を F'_{n-1} とする．ただし，$F'_L = F_L$ とする．このとき，F'_{n-1} は以下のように定式化できる．

$$F'_L = F_L$$
$$F'_i = \mathrm{Merge}(F_i,\ F'_{i+1}) \quad (0 \leq i \leq L-1)$$

統合された特徴マップ F'_0 は，最後に用意された 1×1 の畳み込み層により，X チャネルの尤度マップ H に変換される．ここで X は，セグメンテーション対象となるクラスの総数である．4.4.1 項で述べるように，今回のアプリケーションである腹腔鏡下胆嚢摘出術のセグメンテーションでは，X は 4 に設定されている．最終的に，入力画像中の画素 $(x,\ y)$ が属するクラスは以下のように推測される．

$$\mathrm{class}(I(x,\ y)) = \mathrm{argmax}(H(x,\ y))$$

これまでに説明した再帰的ネットワーク構造により，*Reduce* と *Merge* サブネットワークは，追加のパラメータを要せず異なったスケールに対応した特徴を学んでいくことになる．これは，4.4.2 項で述べる，ランダム反転やリサイズ等のオンラインデータ拡張と組み合わせることで，過学習の危険性を抑え，ネットワークの汎用性を向上させることにつながる．再帰レベル L と分類精度との関係については 4.5.2 項で評価することとする．

この再帰的ネットワーク構造に，本研究ではさらに，G. Huang らにより提案された Stochastic Depth 正則化 [51] を組み合わせることで，よりいっそうの過学習抑制を図っている．Stochastic Depth は，ネットワークの各層を，その深さに依存する一定の確率でバイパスする学習方法である．今回の設計では，2 つのサブネットワーク *Reduce* と *Merge* をバイパスする単位とする．n 個目の *Reduce* または *Merge* がスキップされない確率 p_n を，以下のように定める．

$$p_n = 1 - \frac{n}{L}(1 - p_L) \quad (1 \leq n \leq L) \tag{4.1}$$

ただし，p_L はハイパーパラメータであり，今回は $p_L = 0.5$ としている．したがって，$p_n = 1 - n/(2L)$ である．

図 4.4 は，学習プロセスにおける Stochastic Depth の働きを図示したものである．破線で示されたサブネットワークはバイパスされたことを意味しており，これらは，特徴マップのサイズを一致させるためのアップサンプリングまたはダウンサンプリングを除

き，一切の処理を行わない．なお，詳細な説明は 4.3.4 項に譲る．Stochastic Depth は，Dropout [24] に類似した正則化機構として働き，ネットワークが過学習に陥るのを防ぐ．加えて，Stochastic Depth は実質的な再帰レベルを減少させる効果をもたらすため，学習速度を向上させる．一方推論時には，確率 p_n で起こるバイパスを，p_n によるスケーリングに置き換える．これにより，セグメンテーションの結果は決定論的なものとなる．Stochastic Depth の有無による分類精度の比較は 4.5.2 項で行う．

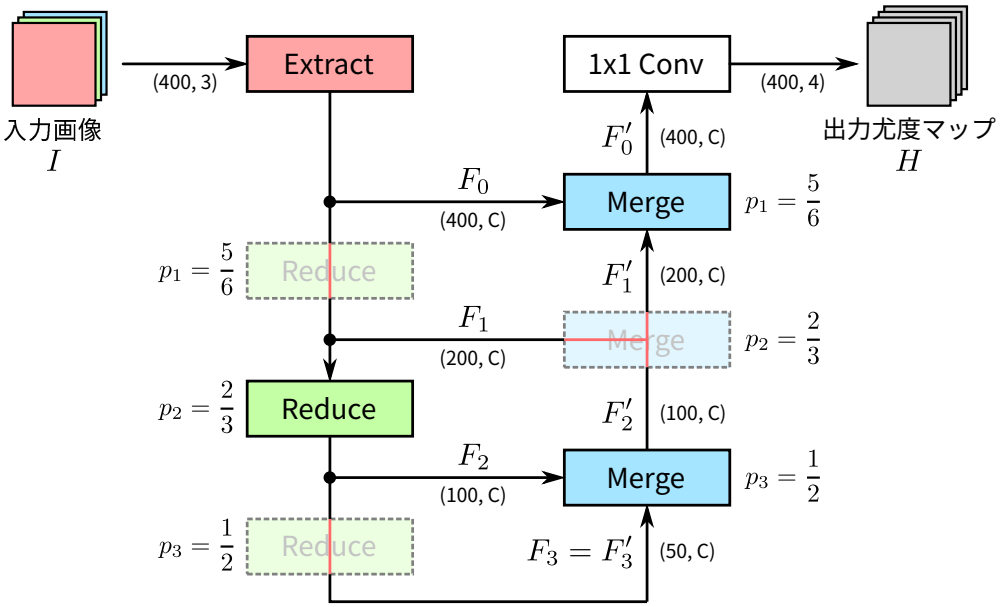
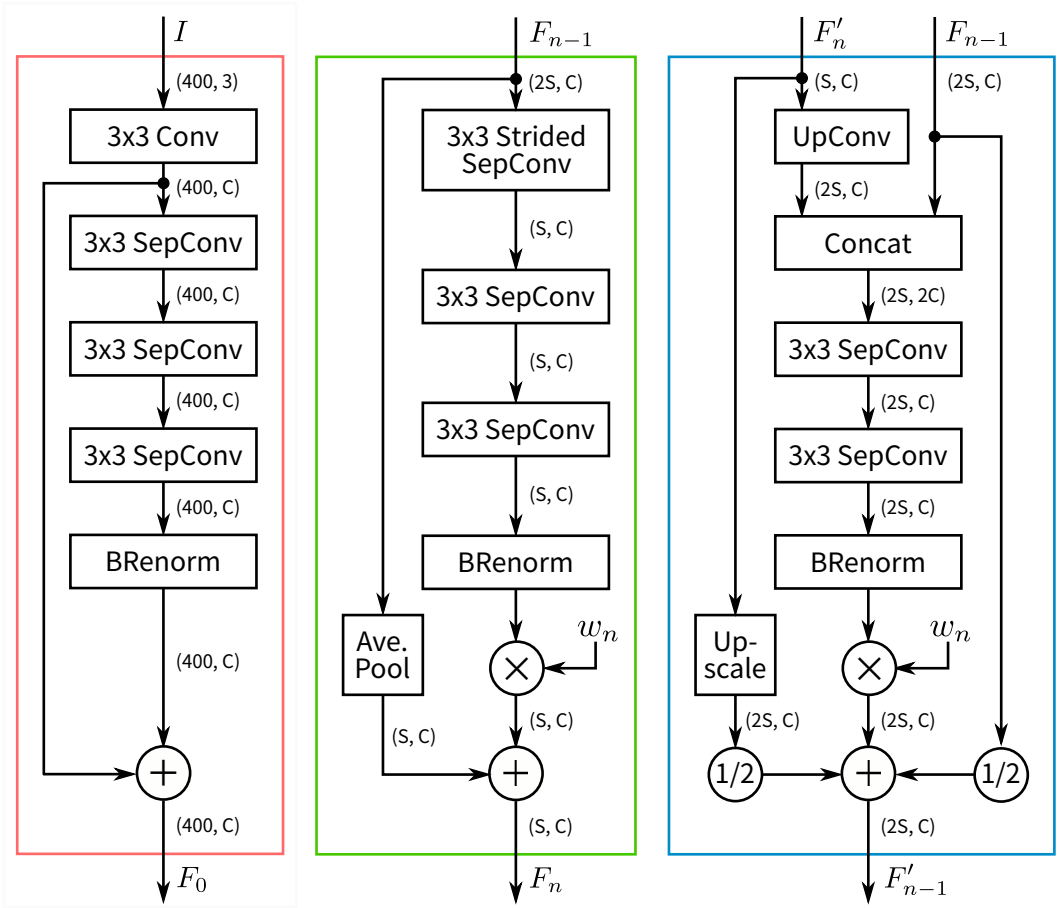


図 4.4: Stochastic Depth に基づくサブネットワークのバイパス

4.3.4　サブネットワーク

各サブネットワークの構成を図 4.5 に示す．*Extract* は 4 つの畳み込み層を備えており，初めの層に限っては，Separable Convolution ではなく，Leaky ReLU を活性化関数とした一般的な 3×3 の畳み込みを行う．これは，入力チャネル数 (図 4.2 における M に相当) が 3 (RGB) と小さく，パラメータ数が小さいためである．C を出力チャネル数 (N) とすると，この最初の畳み込み層は $3^2 \times 3 \times C = 27C$ のパラメータを持つ．一方，残る 3 つの畳み込み層は，3×3 の Separable Convolution を用いており，$M = N = C$ である．したがって，それぞれが持つパラメータ数は $C^2 + 9C$ である．なお，*Extract* に限らず，ネットワーク内の各層では，3×3 畳み込みに先立って幅 1 のゼロパディングが行われるため，入力と出力の特徴マップのサイズは変化しない．これにより，*Merge* における連結が容易に行える．

Reduce や *Merge* も含め，それぞれのサブネットワークにおいて全ての畳み込みが完了すると，Batch Renormalization (BRenorm) [52] が適用される．これは，バッチ正規化 (Batch Normalization) [25] の拡張版で，規模が小さくサンプル間の依存性が高いミニバッチでも，効率的に学習できるように工夫したものである．この特性は，今回用いるデータ



(a) *Extract*　(b) *Reduce*　(c) *Merge*

図 4.5: 各サブネットワークの構成

セットに適している．なお，データセットの詳細については 4.4.1 項にて述べる．また，各サブネットワークには，ResNet [23] に見られるショートカット接続も含まれており，誤差逆伝播法において勾配がスムーズに伝達されるように配慮されている．

Reduce は，3 つの Separable Convolution 層から構成されている．最初の層では，Depthwise Convolution をストライド幅 2 で行うことにより，入力された特徴マップ F_{n-1} をダウンサンプリングしている (Strided SepConv)．3 つの層がそれぞれ持つパラメータ数は，*Extract* と同様に $C^2 + 9C$ である．ストライド幅は影響しない．またショートカット接続部分にも，ダウンサンプリングのため，平均値プーリング (Ave. Pool) が用いられている．ショートカットされた入力値を足す Identity Mapping の前に，Batch Renormalization の適用と，Stochastic Depth 正則化のために用意された重み係数 w_n の乗算がなされる．x を，一様分布 $U(0,\ 1)$ からサンプリングした乱数であるとしたとき，学習プロセスにおける w_n は以下のように決定される．

$$w_n = \begin{cases} 1 & (x \leq p_n) \\ 0 & (\text{otherwise}) \end{cases}$$

ただし，p_n は式 (4.1) で示された，そのサブネットワークがバイパス**されない**確率である．$w_n = 0$ であるとき，サブネットワークの出力は単にダウンサンプリングされた入力値 F_{n-1} と等しく，畳み込み層や Batch Renormalization 層は意味を成さない．実際の実装では，それらは条件分岐により無効化され，演算は一切行われない．推論プロセスにおいては，4.3.3 項で述べたように，$w_n = p_n$ となる．なお，Stochastic Depth は無効化もでき，この場合は学習，推論の別を問わず $w_n = 1$ となる．

他の 2 つのサブネットワークと異なり，*Merge* はサイズの異なる 2 つの特徴マップ F'_n と F_{n-1} を入力とする．前段の *Merge* から入力される小さい特徴マップ (F'_n) に対しては，Separable Convolution を用いたアップサンプリング畳み込み (UpConv) 層が適用され，*Extract* から入力される大きい特徴マップ (F_{n-1}) と連結できるようになっている．このアップサンプリング畳み込み層に対し，本研究では，4.3.5 項で述べる 2 種類のアップサンプリング技法を適用し，評価することとする．いずれの技法を用いた場合でも，パラメータ数は変わらず，$C^2 + 9C$ である．連結された $2C$ チャネルの特徴マップは，続く 2 つの Separable Convolution 層により C チャネルの特徴マップに統合される．この 2 層が持つパラメータ数は，それぞれ $2C^2 + 18C$，$C^2 + 9C$ である．*Merge* もショートカット接続を備えており，F_{n-1} と，ニアレストネイバー法によりアップスケーリングされた F'_n との平均値が結果に加算される仕様となっている．なお，Stochastic Depth による w_n の乗算は *Reduce* と同じである．

表 4.1: サブネットワークごとのフィルタパラメータ数

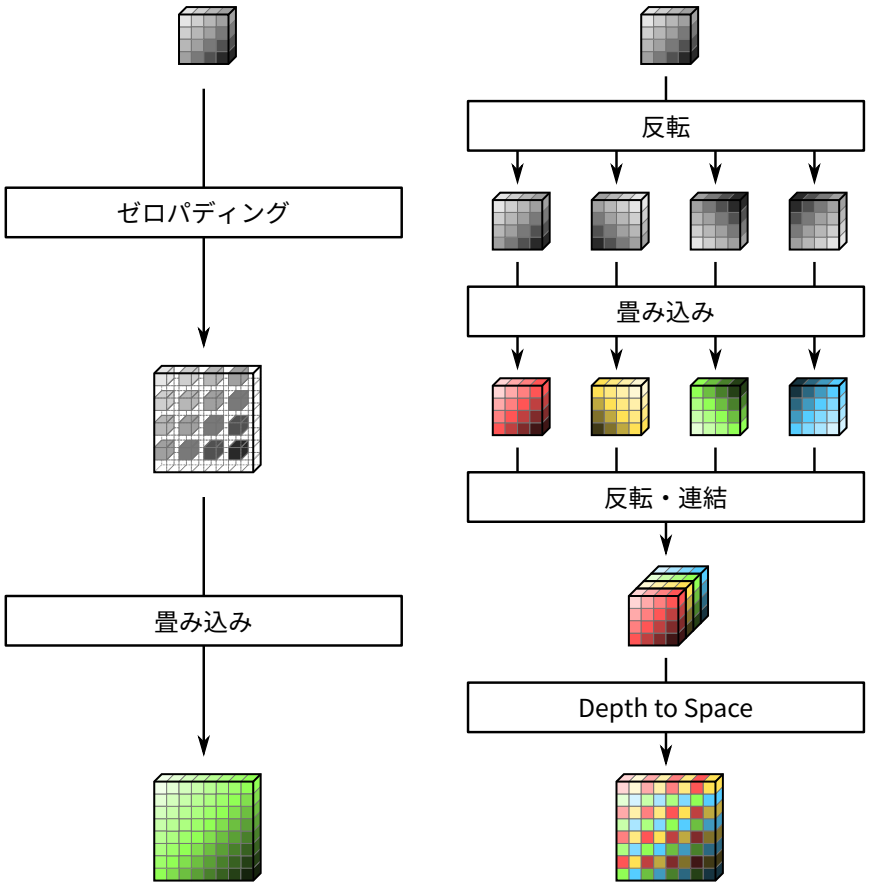
	パラメータ数	
サブネットワーク	SepConv あり	SepConv なし
Extract	$3C^2 + 54C$	$27C^2 + 27C$
Reduce	$3C^2 + 27C$	$27C^2$
Merge	$4C^2 + 36C$	$36C^2$
1×1 Conv	$4C$	$4C$
合計	$10C^2 + 121C$	$90C^2 + 31C$

ネットワーク全体のフィルタパラメータ数を，表 4.1 にまとめる．ここから，Separable Convolution を使用した場合の総パラメータ数は $10C^2 + 121C$ であることが分かる．特記すべき点として，この数字は再帰レベル L の影響を受けない．Separable Convolution を使用しない場合，パラメータ数は $90C^2 + 31C$ に増加する．例として $C = 20$ とすると，Separable Convolution の有無により，システム全体のパラメータ数は 6420 及び 36620 となる．つまり，Separable Convolution の導入により，パラメータ数及び理論上の計算量がおよそ 82.5% 軽減されたことになる．

4.3.5　反転に基づくサブピクセル再構成

Merge サブネットワークにおけるアップサンプリング畳み込み層を実装するにあたり，本研究では，一般的な手法である転置畳み込み (Transposed Convolution) と，反転に基

づくサブピクセル再構成の 2 つの手法を採用し，比較する．両者の概念を図 4.6 に示す．



(a) 転置畳み込み　　(b) 反転に基づくサブピクセル再構成

図 4.6: アップサンプリング手法の比較

転置畳み込みは，逆畳み込み (Deconvolution) や Fractionally-Strided Convolution などとも呼ばれ，畳み込みニューラルネットワークのアップサンプリング層として広く用いられている．図 4.6 (a) に示すように，転置畳み込みを単純に表現すれば，ゼロパディング等による画像拡張に続いて通常の畳み込みを行うものである．本研究では，一般的なゼロパディングを採用する．

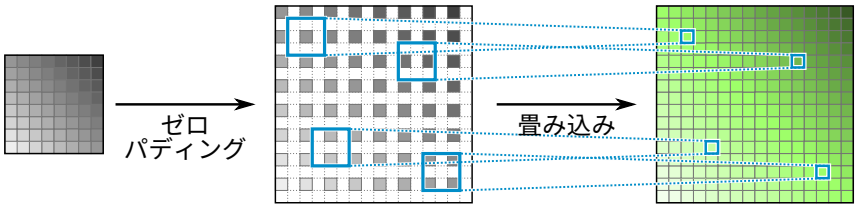
一方の反転に基づくサブピクセル再構成は，第 3 章で提案した超解像システムにおける反転手法 (3.3.2 項) をアップサンプリング層に組み入れたものである．以下にその概要を示す．入力特徴マップのサイズを (C, V, H) とする．ただし，C はチャネル数，V は高さ，H は幅である．まず，入力特徴マップに 3 種類の反転 (水平反転，垂直反転，両反転の併用) を適用する．続いて，オリジナルも含めた 4 種類の特徴マップそれぞれに，同一の畳み込み層を用いた畳み込みを行う．それぞれの結果を，最初と同じ反転を再度適用して本来の向きに戻してから，チャネル方向に連結した後，Depth to Space 変換により，アップサンプリングされた特徴マップを得る．なお，Depth to Space 変換は，3.5.3 項で述べた Space to Depth 変換と対を成すもので，サイズ $(4C, V, H)$ の入力特徴マップ F_{in}

とサイズ $(C, 2V, 2H)$ の出力特徴マップ F_{out} との対応関係は以下のように表される．

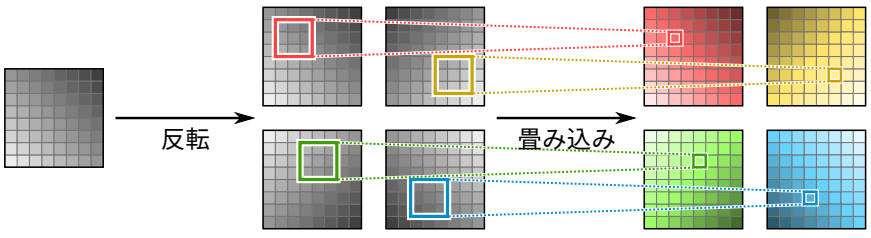
$$F_{\text{out}}(c,\, v,\, h) = F_{\text{in}}\left(c + dC,\, \left\lfloor \frac{v}{2} \right\rfloor,\, \left\lfloor \frac{h}{2} \right\rfloor\right)$$
$$d = (v \bmod 2) \times 2 + (h \bmod 2)$$

ただし、$F(c,\, v,\, h)$ はチャネル c の座標 $(v,\, h)$ における値を意味する．これらの手続きを図示したものが図 4.6 (b) である．Depth to Space 変換のテクニックは，3.2.3 項で述べた超解像システム ESPCN [13] の Pixel Shuffler と同一であるが，入力となる 4 枚の特徴マップを，反転を組み合わせることによりパラメータ数を増やすことなく生成している点が ESPCN と大きく異なる．これにより，反転に基づくサブピクセル再構成を用いたアップサンプリングは，転置畳み込みを用いる場合と同じパラメータ数で実現できる．また，パディングされたゼロに対する乗算も含めた場合，反転や連結等の追加処理を除けば両者の総演算量は同等である．

本研究では，いずれのアップサンプリング手法においても 3×3 の Separable Convolution を用いている．したがって，反転に基づくサブピクセル再構成では，Depthwise Convolution においてフィルタとの畳み込みが行われる画素数はチャネルあたり 9 である．一方転置畳み込みの場合，この画素数は座標によって変化し，1，2 もしくは 4 のいずれかとなる．これは，畳み込みに先立って行われるゼロパディングの影響によるものである．図 4.7 は，この差異を視覚的に表したもので，転置畳み込みでは考慮できる情報量が少なくまた不均一であるのに対し，反転に基づくサブピクセル再構成では，座標によらずフィルタサイズを最大限に活用できることが分かる．両者の違いが品質にもたらす影響については，4.5.2 項で評価する．



(a) 転置畳み込み



(b) 反転に基づくサブピクセル再構成

図 4.7: アップサンプリング手法による畳み込み画素数の違い

4.4　学習

本節では，本研究の手術画像セグメンテーションに用いる畳み込みニューラルネットワークの学習手法について記述する．

4.4.1　データセット

本研究のセグメンテーションシステムは，腹腔鏡下胆嚢摘出術の支援のため，入力画像の各画素を，背景または手術器具 (黒)，胆嚢 (青)，胆嚢管 (黄)，総胆管 (赤) の 4 クラスに分類するように設計されている．以下，それぞれのクラスを可視化する際には，括弧内に示した色を用いるものとする．4.3 節で述べたネットワークの学習に用いる学習データセットは，ネットワークに与える入力画像と，推論された分類結果と比較するための教師ラベル画像の組から構成され，学習用として 138 組，評価用として 45 組が用意されている．データセットに含まれる画像の例を図 4.8 に示す．画像サイズは 640×512 である．*Merge* サブネットワークが正常に動作するために，幅と高さはいずれも 2^L の倍数でなければならない．ただし，L は再帰レベルである．

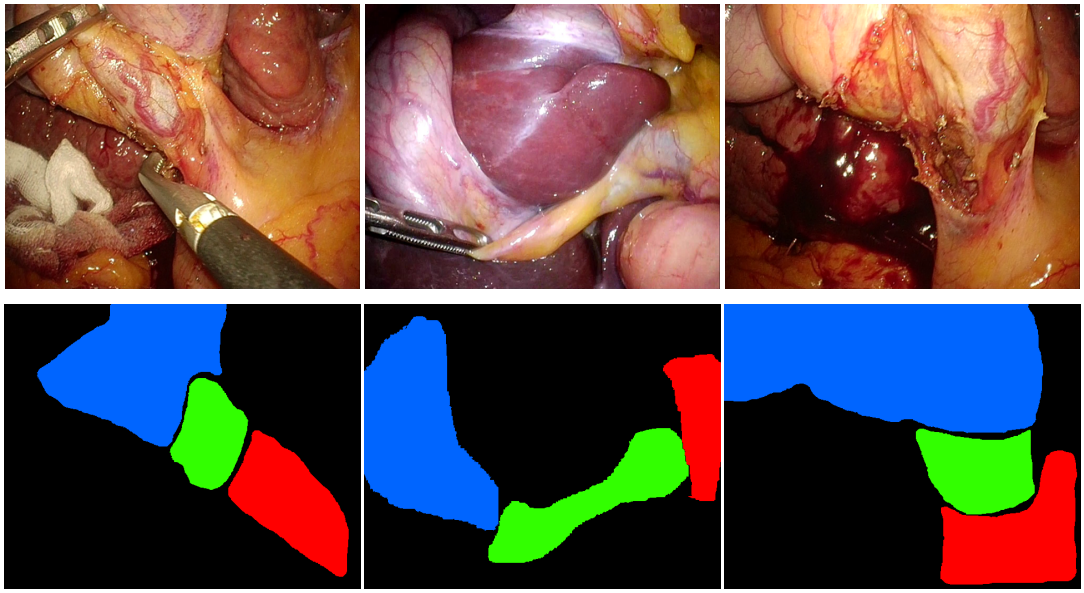


図 4.8: 学習データセットに含まれる入力画像と教師ラベル画像の例

全ての入力画像は，57 人の患者に対して行われた実際の腹腔鏡下胆嚢摘出術の映像からキャプチャしたものであり，対応するラベル画像は，外科医の手でアノテーションされたものである．ラベル画像は色で可視化されているが，実際には，各クラスを指し示す番号が並べられた配列として扱われる．アノテーションは知識と経験を要する難しい作業であり，炎症の激しい症例では特に，熟練した外科医であっても時間がかかる．また，プライバシーの観点から画像の取得と管理にも十分な配慮が求められる．このような事情から，一般の深層ニューラルネットワークで用いられているような，数千から数万の組を含むものと比較して，現時点でのデータセットの規模は極めて限定的なものとなっている．4.3

表 4.2: データセットにおける各クラスの分布

クラス	表示色	学習	評価	合計
背景・器具	黒	65.46%	64.97%	65.34%
胆嚢	青	21.37%	22.35%	21.61%
胆嚢管	緑	4.96%	5.49%	5.09%
総胆管	赤	8.21%	7.18%	7.96%
画像の組の数		138	45	183

節で過学習の抑制を意図したネットワーク構造を導入したが，これと併せて，データセット規模のさらなる改善も引き続き行う予定である．

クラス分類を扱うデータセットでは珍しいことではないが，データセット内で各クラスが占める割合には大きな不均衡がある．各クラスの分布は表 4.2 に示すとおりであり，背景・器具のクラスに属する画素が全体の約 65% を占める一方，胆嚢管の割合はわずか 5% 前後である．そのため，分類精度の評価にあたっては，この不均衡を考慮に入れなければならない．そのための評価手法については，4.5.1 項で述べることとする．

4.4.2　ツールと学習手順

ネットワークを定義し学習を進めるにあたり，本研究では，第 3 章の超解像システムと同様に Chainer [39] 5.3.0 を利用する．また，訓練や評価で必要となる画像処理についても，同様に OpenCV-Python [40] を利用して行う．

学習手順は以下に示すとおりである．まず初めに，学習データセットから画像の組を 2 つ読み出す．続いて，ランダムな座標から小画像を切り出し，そのそれぞれについてデータ拡張のため，以下に列挙する 3 種類の加工を施す．

- **回転**．`cv2.warpAffine()` 関数を用いて，一様分布 $U(0,\ 2\pi)$ からサンプリングしたランダムな角度による回転を行う．入力画像は，ジャギーの発生を抑えるため Lanczos4 [53] アルゴリズムで補間 (`cv2.INTER_LANCZOS4`) するが，ラベル画像は 4 値しか取れないため，最近傍補間 (`cv2.INTER_NEAREST`) を用いる．なお，画像の範囲外にあたる部分の画素値は 0 とする．

- **水平反転**．0.5 の確率で，水平方向の反転を加える．なお，垂直反転と水平反転の併用は π による回転と等価であるため，水平方向の反転のみで十分である．

- **リサイズ**．`cv2.resize()` 関数を用いて，一様分布 $U(2/3,\ 4/3)$ からサンプリングしたランダムな倍率 s によるリサイズを行う．この分布範囲は，ネットワーク内で 2 倍単位でのリサンプリングが行われることを考慮し，最小と最大の比が 2 となるように設定している．補間アルゴリズムは回転と同様に，入力画像は Lanczos4，ラベル画像は最近傍補間である．実際の処理では，小画像切り出し時のサイズを $(400/s)\times(400/s)$ と定め，最終的に 400×400 にリサイズすることで，$400/(400/s) = s$ の拡縮倍率が得られる仕組みとなっている．s の値域から，切り出しサイズは 300×300 から

600×600 の範囲内となる．切り出しサイズがデータセットの画像サイズ 640×512 を超える場合，範囲外にあたる画素値は 0 とする．

これらのデータ拡張はオンラインで行われる．言い換えると，事前にデータセットに対して適用しておくのではなく，学習のプロセスと並行して行われるということである．このため，データセットが肥大化せず，全データをメモリに保持して学習が進められる．また，回転角度及びリサイズの倍率を浮動小数点数で柔軟に決定できるため，より多様で効果的なデータ拡張が実現できる．

さらに本研究では，追加のデータ拡張として，2.5 節で述べた PCA Color Augmentation の併用も検討している．ただし，第 3 章の超解像システムとは異なり，腹腔鏡に搭載されたカメラは，ホワイトバランスや明るさを十分に調整しながら利用されるものであるため，画素値に外乱を加えることが必ずしも適当ではない可能性がある．その一方で，自律的な腹腔鏡制御を行う場合，照明が腹腔鏡の先端に取り付けられており，移動に伴って照明環境が著しく変化するため，実際の手術映像であるデータセットには含まれない不適切な明るさの映像が得られることも想定され，そういった場合に PCA Color Augmentation が有効に働く可能性もある．したがって，利用の有無により品質がどのように変化するかを比較，評価することとした．その結果は 4.5.2 項で述べる．

これまでに述べたデータ拡張の処理により，400×400 の入力画像及び教師ラベル画像の組が 2 つ生成される．これをミニバッチとして，ネットワークの学習を進める．ミニバッチの生成手順を図 4.9 にまとめる．誤差関数にはソフトマックス交差エントロピーを利用し，最適化関数には Adam [42] をデフォルトパラメータ ($\alpha = 0.001$，$\beta_1 = 0.9$，$\beta_2 = 0.999$，$\epsilon = 10^{-8}$) で利用している．データセットの全画像の学習が完了すると，順序をシャッフルした上で再び最初から学習を続けていく．

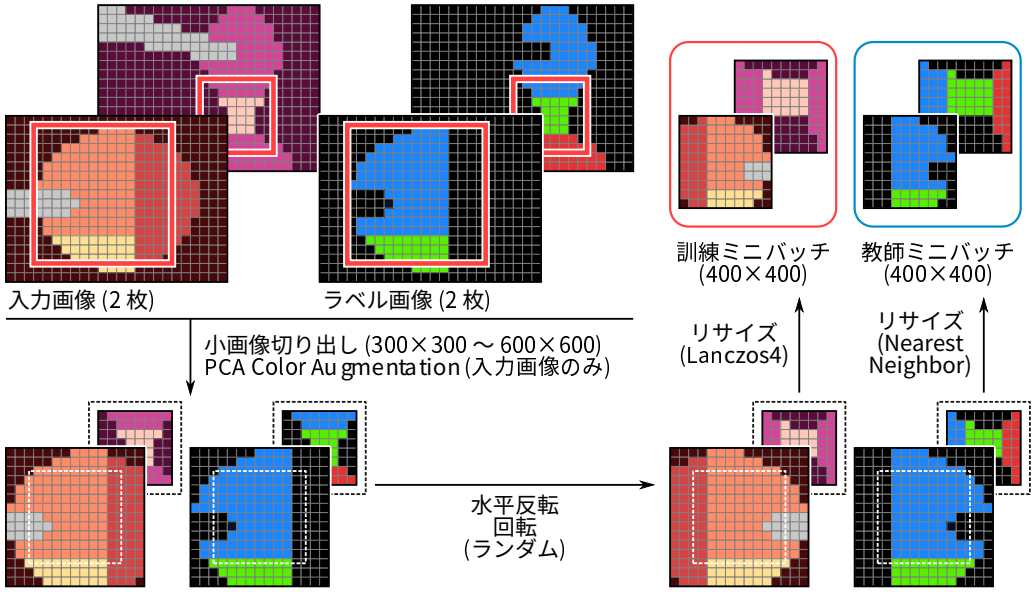


図 4.9: ネットワークの学習の流れ

4.5　評価と考察

本節では，各種のネットワーク構成やアップサンプリング手法によるセグメンテーションの分類精度の変化を評価，考察する．また，GPU を用いた推論速度の評価も併せて行う．

4.5.1　評価手法

分類精度の評価指標として，本研究では以下の式により定義される Mean Accuracy を用いる．

$$\text{Mean accuracy} = \frac{1}{X}\sum_{i=0}^{X-1}\frac{n_i}{t_i}$$

ここで，$X = 4$ は分類するクラス数であり，t_i は教師ラベル画像においてクラス i に属する画素の総数，n_i は，クラス i に属する画素のうち，正しくクラス i に分類された画素の総数である．より単純な評価指標である Pixel Accuracy $(\sum n_i)/(\sum t_i)$ では，広範囲を占めるクラスの影響を過度に受けてしまうが，Mean Accuracy は割合の少ないクラスに対する推論精度を適切に反映できる．今回用いるデータセットでは，表 4.2 で示したように，評価データセットの全画素中 65.0% が背景・器具クラスに属している．このため，もしシステムが全ての画素を一律に背景へ分類しても，Pixel Accuracy による分類精度は 65.0% と高い値になってしまう．一方 Mean Accuracy を用いた場合，分類精度は 25.0% となり，より適切な評価が可能となる．

4.3 節で述べた設計上の工夫と，4.4 節で述べた PCA Color Augmentation の有効性を評価するため，表 4.3 にまとめた計 12 のシステム構成において，Mean Accuracy に基づいた分類精度を求める．なお，Flip，Trans はそれぞれ，4.3.5 項で示した反転に基づくサブピクセル再構成と転置畳み込みを表しており，SD は Stochastic Depth，CA は PCA Color Augmentation の略である．また，C は各サブネットワークの出力チャネル数であり，L は再帰レベルを指す．C は Flip30 L4+SD 構成のみ $C = 30$ とし，他の構成は全て $C = 20$ としている．加えて，再帰構造を持たない一般的なネットワーク構造と，本研究のモデルを比較するため，FlipNR L4+SD 構成も評価対象に加えている．これは，*Reduce* と *Merge* サブネットワークを再使用せず，独立したパラメータを持った別個のネットワークとして配置したものである．

4.4 節で説明した学習のプロセスにおいて，1000 ミニバッチの学習 (以降イタレーションと呼ぶ) が完了するたびに，評価データセットを使った分類精度の評価を行う．つまり，n イタレーションの学習が完了したということは，$1000n$ ミニバッチ，言い換えると 400×400 の入力画像と教師ラベル画像が計 $2000n$ 組ネットワークに与えられて，パラメータが更新されたということを意味する．

4.5.2　分類精度

まず，複数の再帰レベルと Stochastic Depth 正則化の有無を組み合わせた複数の Flip 構成 (Flip L1, L2, L2+SD, L4, L4+SD) で分類精度を評価する．図 4.10 が，それぞれの

表 4.3: 評価したシステム構成の一覧

名称	UpConv	再帰構造なし	C	L	Stochastic Depth	PCA Color Augmentation
Flip L1	Flip	-	20	1	-	-
Flip L2		-	20	2	-	-
Flip L2+SD		-	20	2	Yes	-
Flip L4		-	20	4	-	-
Flip L4+SD		-	20	4	Yes	-
Flip L4+CA		-	20	4	-	Yes
Flip L4+SD+CA		-	20	4	Yes	Yes
Flip30 L4+SD		-	30	4	Yes	-
FlipNR L4+SD		Yes	20	4	Yes	-
Trans L1	Trans	-	20	1	-	-
Trans L2+SD		-	20	2	Yes	-
Trans L4+SD		-	20	4	Yes	-

構成における学習中の分類精度の変化を表している．なお，視認性を確保するため，3.6 節と同様，近似曲線による表示となっている点に注意されたい．この結果から，再帰レベル L を大きくし，さらに Stochastic Depth を組み合わせることで，より優れたピーク精度が得られると言える．L が大きくなると，ネットワークがそれぞれの出力画素を生成するために参照する入力画像中の領域，つまり受容野が広がる．詳細な評価は 4.5.3 項で行うが，L が 1 増加するたびに，受容野の大きさは概ね縦横 2 倍となる．大きい L を使用することで，パラメータ数に影響を及ぼすことなく受容野が拡大され，大局的なコンテキストを考慮できるようになるため，分類精度の向上に貢献すると考えられる．

ただし，L が大きくなると，分類精度が比較的早期にピークに達し，その後減少に転じる傾向が見られる．図 4.10 を見ると，この現象は $L = 4$ の構成で顕著であることが分かる．ピークに達した後でも，学習中のソフトマックス交差エントロピー誤差に増加は見られなかったことから，これは過学習の発生を示唆するものである．この精度低下は，Stochastic Depth の併用により抑制されており，Stochastic Depth が過学習の軽減に有効であることを示している．一方，$L = 1$ の構成では，パラメータ数が同等であるにもかかわらず精度低下が見られない．考えられる要因としては，受容野が小さいため，データセットへの過剰な適応が難しくなったということが挙げられる．しかしながら，$L = 1$ における分類精度は比較対象中で最低であるため，この再帰レベルを採用すべき理由はない．

次に，PCA Color Augmentation がもたらす影響について評価を行う．図 4.11 は，Flip L4 と L4+SD 構成を用いて，PCA Color Augmentation の有無による分類精度の変化を比較したグラフである．このグラフから，PCA Color Augmentation を併用した場合，過学習の傾向は抑えられるものの，ピークの精度について言えば L4 と L4+SD いずれの構成においても低下していることが分かる．このことは，ネットワークが臓器を認識するにあたり，特定の色の情報にある程度依存していることを示している．

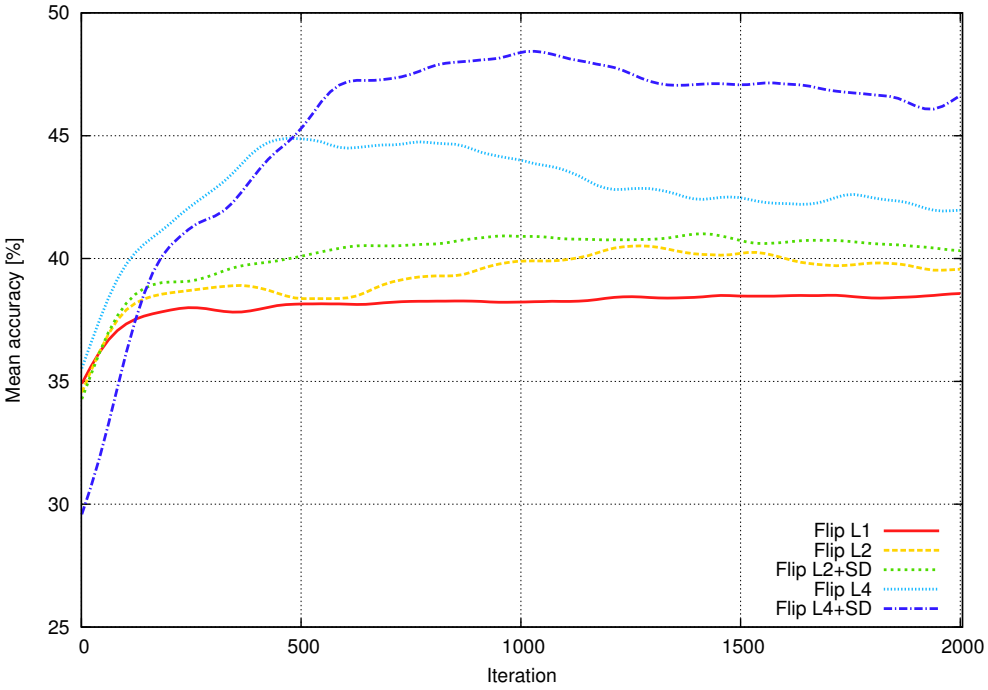


図 4.10: Flip 構成の再帰レベルによる分類精度の比較

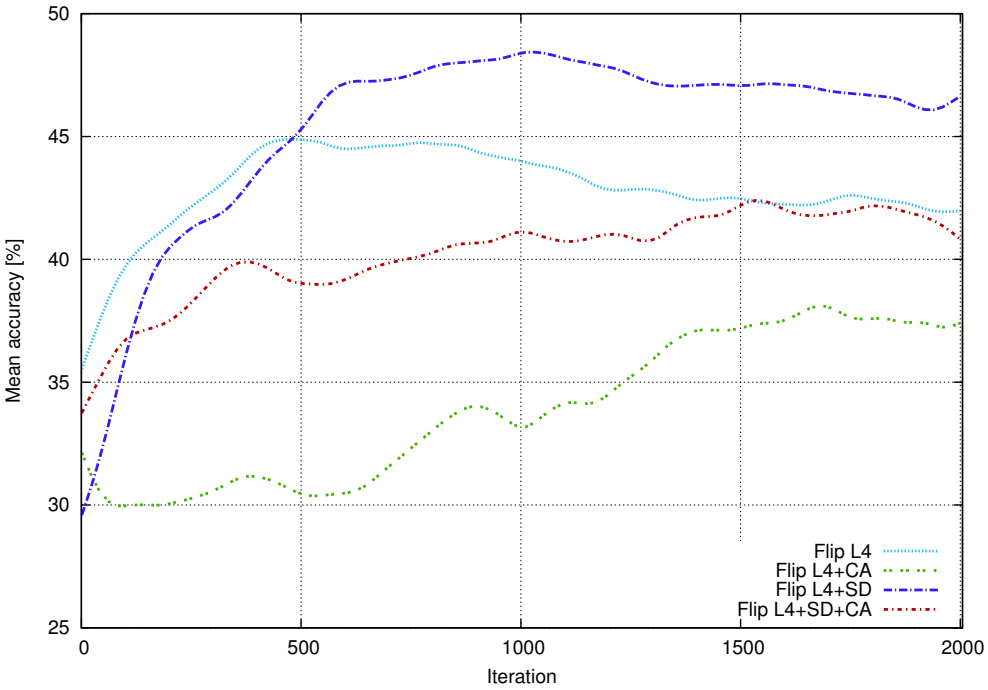


図 4.11: Flip 構成の PCA Color Augmentation の有無による分類精度の比較

データセットに含まれている画像は，4.4.1 項で述べたように実際の手術映像からキャプチャしたもので，ホワイトバランスや明るさ，照明条件が適切にコントロールされているため色情報を利用しやすいが，これを PCA Color Augmentation が乱してしまうため，分類がより難しくなったと考えられる．ただし，実際の腹腔鏡自動制御にあたっては，腹腔鏡とともに光源が移動することで発生する複雑な照明条件の変化や，個人によって異なる臓器の色味の違いなどに対応することが求められ，このような環境下では，PCA Color Augmentation が有効性を示す可能性もある．これを検証するためには，より多様性に富んだ大規模なデータセットが必要である．

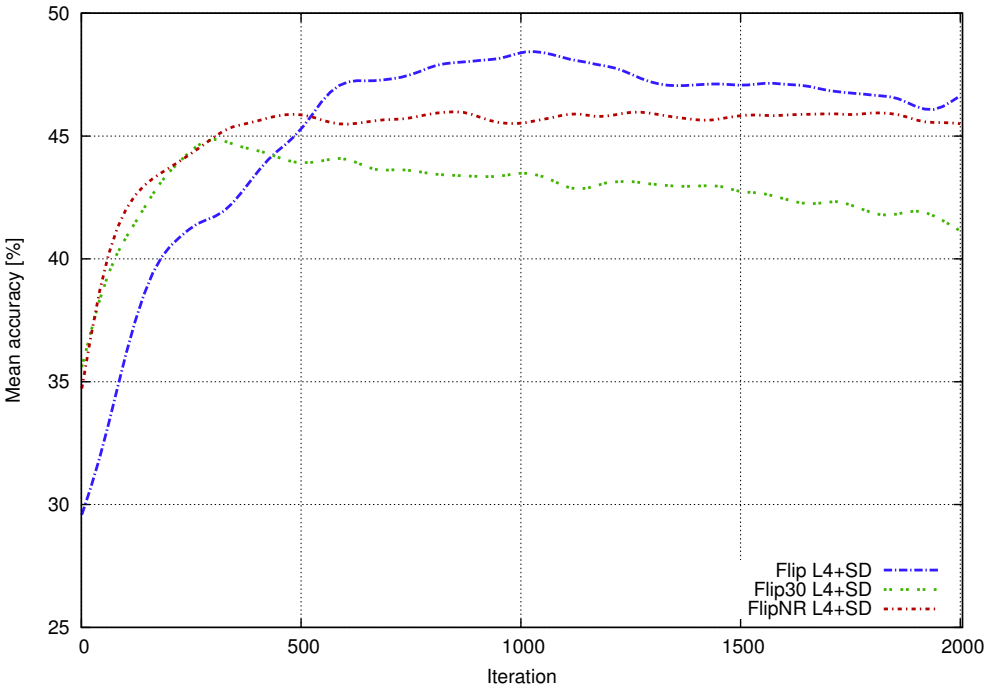
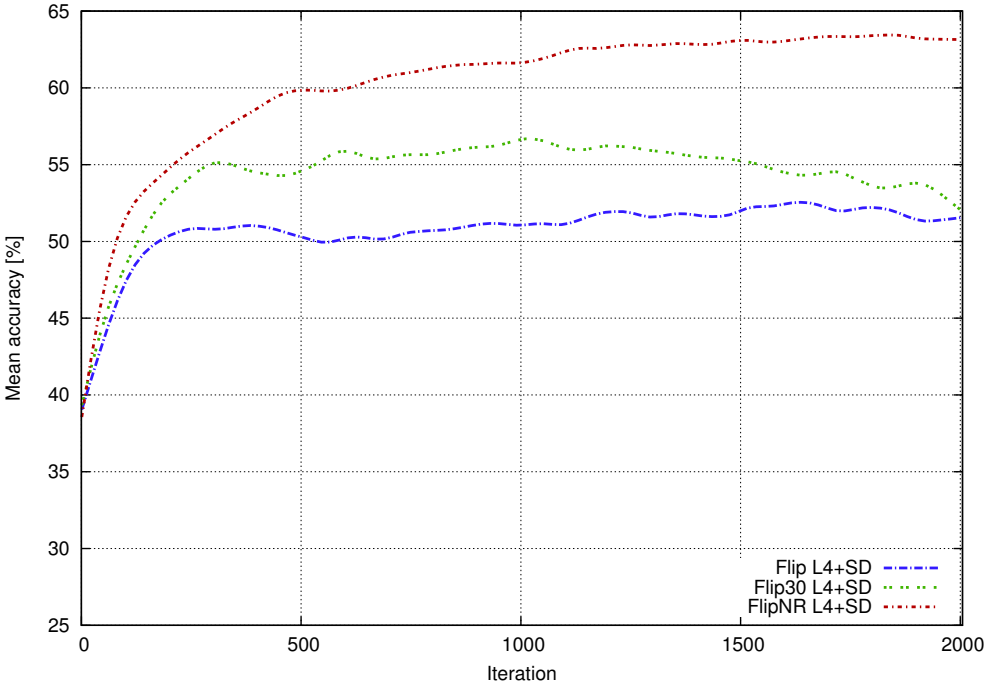
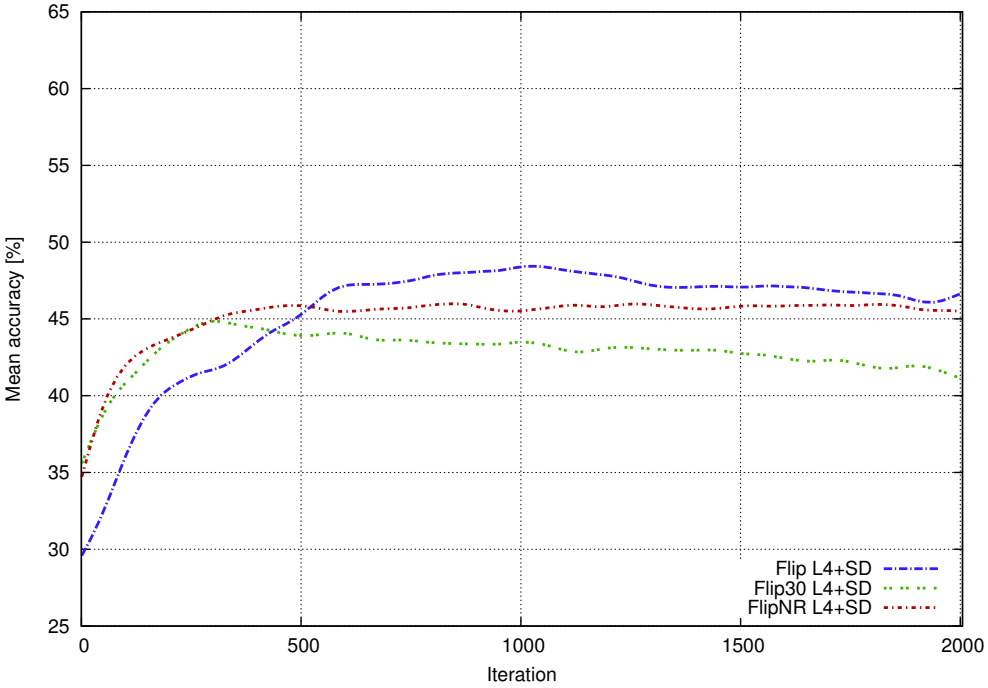


図 4.12: Flip，Flip30，FlipNR 構成 (L4+SD) の分類精度の比較

続いて，本研究で導入した再帰的ネットワーク構造と，チャネル数 C が分類精度に与える影響を評価するため，L4+SD を採用した Flip (パラメータ数 6640)，Flip30 (パラメータ数 12630)，FlipNR (パラメータ数 18600) の 3 構成で分類精度を比較した．その結果が図 4.12 である．これを見ると，最もパラメータの少ない Flip L4+SD が，他の 2 構成より優れた精度を達成していることが分かる．一方で Flip30 は早期に精度がピークに達し，その後継続的に減少している．再帰構造を含まない FlipNR は品質の推移が比較的安定しているが，分類精度は Flip に及ばない．このパラメータ数と品質の逆転現象は，過学習を強く示唆している．そこで確認のため，学習データセットと評価データセットでの品質の差異を比較した．その結果を図 4.13 に示す．学習データセットを用いた場合 (図 4.13 (a))，パラメータ数が多い構成ほど高いピーク精度を実現しており，特に FlipNR の品質は 60% 超に達している．しかし，評価データセットでの評価 (図 4.13 (b)) では，FlipNR の分類精度は大きく低下し，Flip を下回っている．これは，FlipNR が学習データセットに過剰に適応していることを意味する．



(a) 学習データセットでの品質



(b) 評価データセットでの品質

図 4.13: 学習データセットと評価データセットの分類精度の比較

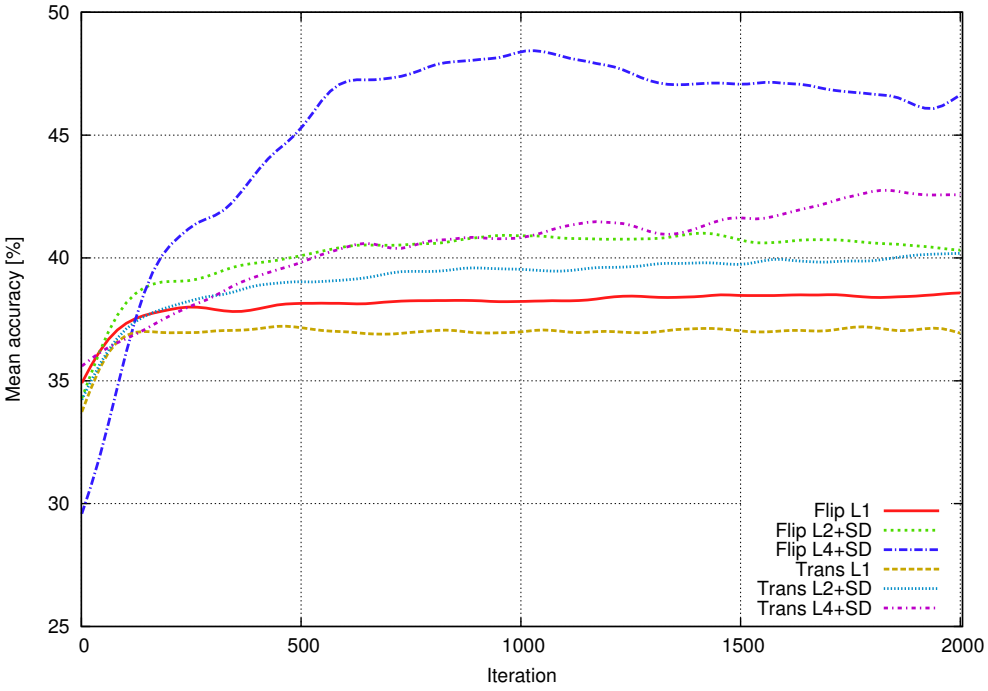


図 4.14: Flip 構成と Trans 構成での分類精度の比較

最後に，L1，L2+SD，L4+SD を用いた Flip と Trans 構成の比較を行い，2 つのアップサンプリング手法が分類精度に与える影響を評価する．その結果は図 4.14 に示すとおりであり，いずれの手法においても L が大きくなるにつれて精度は向上しているが，同一の再帰レベルでは Flip の精度がより優れていることが分かる．特に $L = 4$ の場合において差が大きい．4.3.5 項で比較したように，転置畳み込みはゼロパディングの後に行われるため，畳み込みに利用できる画素数が 1, 2 または 4 と少なく，また不均一である．9 画素を全て利用できる反転に基づくサブピクセル再構成と比較すると，転置畳み込みは表現力が低く，これが品質差につながっていると考えられる．また，両者は受容野の大きさという面でもわずかに差がある．この点については 4.5.3 項で議論する．

以上の評価から，再帰的なネットワーク構造を Stochastic Depth 正則化，反転に基づくサブピクセル再構成によるアップサンプリングと併せて導入することにより，パラメータ数を増加させることなく，分類精度が向上することが示された．一方で PCA Color Augmentation については，今回のデータセットに基づく分類タスクでは有効ではなかった．ただし，実際の利用環境下では有効である可能性はあるため，データセットの改善を通して検証する必要がある．

評価した全システム構成における，2000 イタレーションまで学習を行った際のピーク分類精度を表 4.4 にまとめる．なお，これまでに示した分類精度のグラフは近似曲線であるため，表 4.4 の数値はグラフより大きく見えるという点に注意されたい．ピーク分類精度の値は，これまでの評価と一致する傾向を示しており，Flip L4+SD 構成が最も高い 55.1% の評価精度を達成している．また，図 4.15 に，表 4.4 の各構成を用いた実際のセグメンテーション結果の例をいくつか示す．$L = 1$ の場合，それぞれのクラスが細かく散

表 4.4: 各システム構成におけるピーク分類精度

Name	Mean Accuracy (%)	イタレーション
Flip L1	40.7	1905
Flip L2	43.6	1336
Flip L2+SD	43.3	1436
Flip L4	52.3	910
Flip L4+SD	**55.1**	1059
Flip L4+CA	46.8	1906
Flip L4+SD+CA	49.3	1912
Flip30 L4+SD	53.9	266
FlipNR L4+SD	53.7	326
Trans L1	39.6	802
Trans L2+SD	42.5	1468
Trans L4+SD	45.5	1748

らばっているが，$L = 4$ ではより組織だった結果が得られていることが分かる．特に Flip L4+SD 構成では，3 種類の臓器の配置が概ね正しく検出されている．画素単位のセマンティックセグメンテーションとしては依然として改善の余地が大きいものの，腹腔鏡の自動制御というアプリケーションを考えると，一定の実用性がある結果とみなせる．

4.5.3　受容野

4.5.2 項で示した再帰レベル L による分類精度の変化には，受容野の大きさが関係していると考えられる．そこで本項では，各構成における受容野の大きさを比較する．

システム全体の受容野を考える前に，まず，サブネットワーク単位での受容野を見ていく．各サブネットワークの出力特徴マップのサイズを r (つまり，$r \times r$) としたとき，その特徴マップの生成に関与する入力特徴マップ中の領域，すなわち受容野のサイズは図 4.16 のようになる．(a) *Extract* はリサンプリングを行わず，3×3 の畳み込みを 4 回行うだけであるため，受容野のサイズは容易に求められ，$r + 8$ である．$r = 1$ とすると，$r + 8 = 9$ となる．つまり，*Extract* の出力特徴マップの各画素は，入力 RGB 画像中の 9×9 画素から生成される．(b) *Reduce* は，幅 2 のストライドを用いたダウンサンプリングを行うため，*Extract* よりも受容野が大きく，そのサイズは $2r + 9$ である．したがって，*Reduce* の出力 1 画素は 11×11 画素から生成される．*Merge* の受容野は，採用するアップサンプリング手法により若干変化する．具体的には，(c) 転置畳み込みを用いる場合のサイズは $\lceil r/2 \rceil + 3$ であるのに対し，(d) 反転に基づくサブピクセル再構成を用いる場合のサイズは $\lceil r/2 \rceil + 4$ と，1 画素分大きい．これは，転置畳み込みにおけるゼロパディング後の 3×3 畳み込みが，実質的には 1×1 から 2×2 の畳み込みに相当するためである．このことからも分かるように，転置畳み込みにおける受容野のサイズは座標により変動するが，今回は最大となる場合の値としている．

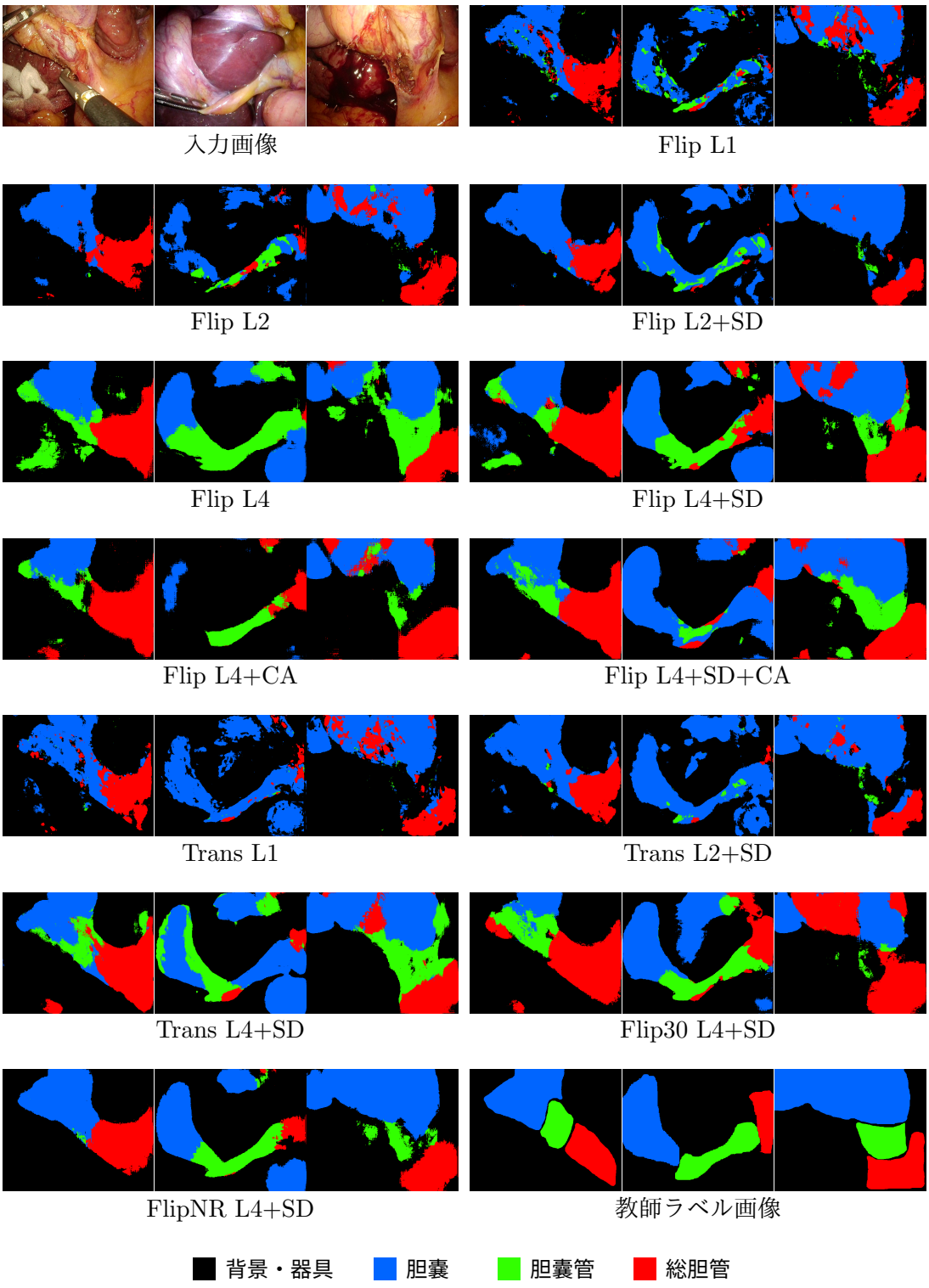
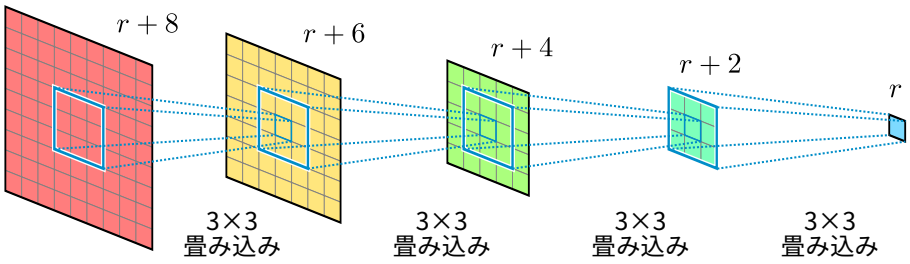
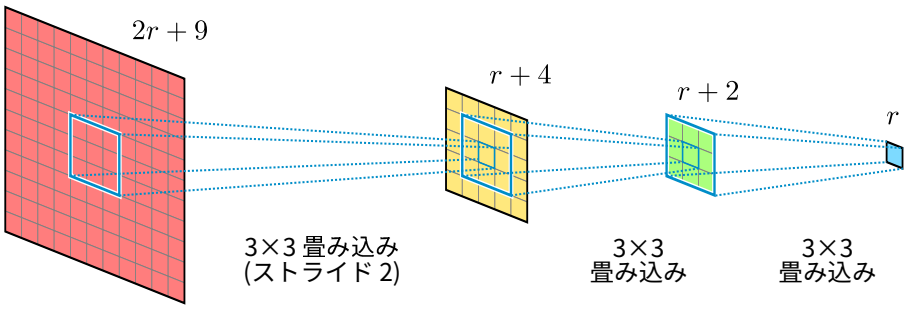


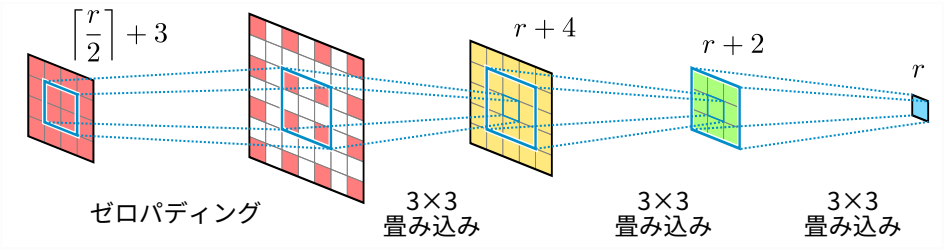
図 4.15: セグメンテーション結果の例



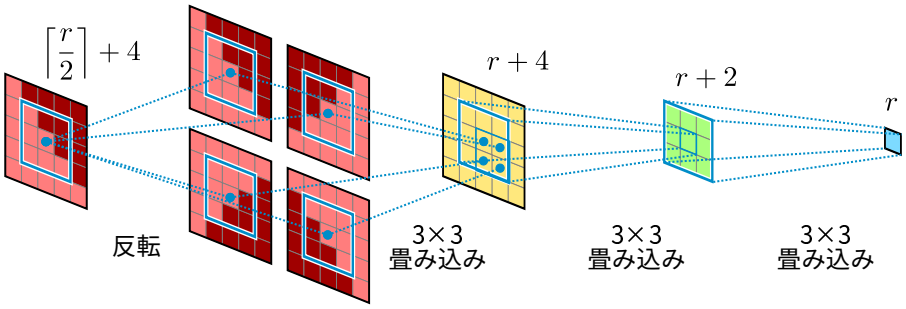
(a) *Extract* の受容野



(b) *Reduce* の受容野



(c) 転置畳み込みを用いた *Merge* の受容野



(d) 反転に基づくサブピクセル再構成を用いた *Merge* の受容野

図 4.16: 各サブネットワークの受容野の大きさ

表 4.5: 再帰レベル L の変化による受容野の大きさの変化

L	1	2	3	4
Flip	27	63	135	271
Trans	25	55	119	239

以上を踏まえて，システム全体の受容野の大きさを考える．4.3.3 項の図 4.3 から分かるように，本システムのネットワークには局所的な細かい特徴を出力に反映させるためのスキップ接続があるため，出力から入力までの経路は複数存在する．そこで，受容野の計算にあたっては，最も長い経路，すなわち再帰利用される全てのサブネットワークを漏れなく通過する経路を考えることとする．このとき，出力尤度マップの各画素に対応する入力 RGB 画像中の受容野のサイズは，r の初期値を 1 とし，先述したサブネットワークごとのサイズ計算式を繰り返し適用することで求められる．例として，転置畳み込みを用いた構成で $L = 3$ の場合，

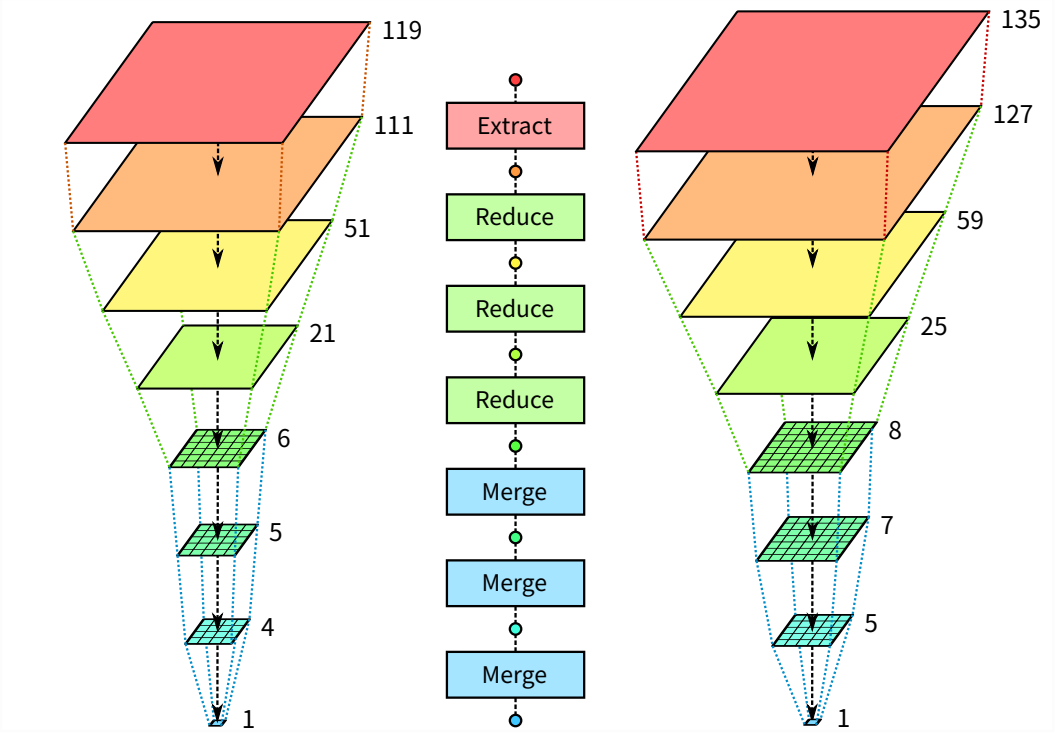
$$
\begin{aligned}
r_{M1} &= \lceil 1/2 \rceil + 3 = 4 \\
r_{M2} &= \lceil 4/2 \rceil + 3 = 5 \\
r_{M3} &= \lceil 5/2 \rceil + 3 = 6 \\
r_{R1} &= 2 \times 6 + 9 = 21 \\
r_{R2} &= 2 \times 21 + 9 = 51 \\
r_{R3} &= 2 \times 51 + 9 = 111 \\
r_{E} &= 111 + 8 = 119
\end{aligned}
$$

となる．ただし，r_{Mi}, r_{Ri} はそれぞれ *Merge*，*Reduce* を i 回通過した時点，r_E は *Extract* を通過した時点での受容野の大きさである．つまり，ネットワーク全体としての受容野の大きさは 119 (119×119) となる．この計算過程を図示したものが図 4.17 である．また，同様の計算により算出した，$L = 1,\ 2,\ 3,\ 4$ の受容野の大きさを表 4.5 にまとめる．ここから，L が 1 大きくなるたびに受容野の大きさが 2 倍程度に増大すること，反転に基づくサブピクセル再構成を用いたほうが転置畳み込みよりも受容野が大きくなることが読み取れる．このことは，4.5.2 項の評価において Flip 構成が Trans 構成よりも高い品質を実現できた一因と考えられる．

4.5.4　推論速度

GPU を用いた場合の推論速度を，デスクトップ PC (OS: Ubuntu 14.04, RAM: 64 GB, CPU: Intel Core i7-5930K，GPU: NVIDIA GeForce GTX 1080) を用いて評価した．ここで用いるシステム構成は，4.5.2 項での分類精度評価において最良の結果を達成した Flip L4+SD とする．なお，推論は，学習時と同様に Chainer を用いて行う．

プロトタイプの腹腔鏡操作ロボットで用いられているカメラの解像度に合わせ，640×480 画素 (VGA) の動画ファイルを入力として評価を行ったところ，約 17 fps (frames per



(a) 転置畳み込み　　(b) 反転に基づくサブピクセル再構成

図 4.17: $L = 3$ の場合におけるシステム全体での受容野の大きさ

second) のフレームレートでセグメンテーションが実行できることが分かった．Flip L4 とほぼ同等の構成で，Separable Convolution の代わりに通常の 3×3 畳み込みを用いた場合，理論上の計算量はより大きいにもかかわらず，40 から 50 fps が得られることを確認している．このような速度低下が見られる理由として，今回の実装では，Chainer の `chainer.links.Convolution2D()` クラスにおいて，groups 引数に入出力チャネル数と同じ数を与えることにより Depthwise Convolution を実現しているが，この部分の実装で処理にオーバーヘッドが生じていることが考えられる．

しかしながら，腹腔鏡は頻繁に動き回るものではないことから，17 fps というフレームレートは，腹腔鏡の自律制御というアプリケーションには十分であると考えられる．実際，実働するロボットと組み合わせた検証実験を何度か行っているが，腹腔鏡はスムーズに動作しており，フレームレートをさらに上げる必要性は見いだされなかった．

4.6　総括

本章では，自律的な腹腔鏡制御を実現するための手術画像セグメンテーションシステムに向けた畳み込みニューラルネットワークのアーキテクチャを提案した．このアーキテクチャは，Depthwise Separable Convolution を用いたエンコーダ・デコーダ型構造を採用しており，内包するサブネットワークを再帰的に利用する再帰構造と，反転に基づくサブピクセル再構成を用いたアップサンプリング手法の導入が大きな特徴である．また，この再

帰構造に Stochastic Depth 正則化を組み合わせている．これらの工夫により，パラメータ数を増やすことなくピークの分類精度を向上させることができ，小規模な学習データセットという制約の中で，55.1% の Mean Accuracy を達成した．また，ランダムなリサイズ，回転，反転といった基本的なオンラインデータ拡張に加えて，PCA Color Augmentation の追加による効果も併せて検証した．その結果，ピークの分類精度は低下したものの，学習の進行に伴う精度の低下には一定の抑制効果が見られ，実利用環境下において有効である可能性が示唆された．システムはシングルの NVIDIA GeForce GTX 1080 GPU を用いて 17 fps で動作することが確認されており，これはロボットによる腹腔鏡制御には十分なパフォーマンスである．

今後の課題として，分類精度をさらに向上させるとともに，より実用的な評価を行うために，データセットの強化が必要である．また，それと並行して，小さなデータセットに合わせて設計されている現在のネットワーク構造を，大規模なデータセットに対応するように改良していくことが求められる．本研究の再帰的なネットワーク構造は，様々なテクニックと組み合わせることができる．例として，Squeeze-and-Excitation Network (SENet) [45] や Context Encoding Network (EncNet) [54] では，パラメータの大幅な増加なく大局的なコンテキストを考慮する仕組みが提案されている．本研究のセグメンテーションシステムは特定の腹腔鏡手術をターゲットとしており，このようなテクニックは，3 つの臓器の配置に関する制約といった，データセットに内在する規則性を学習する上で有効であると考えられる．それにより，現実にはありえないセグメンテーション結果が出力されるのを防ぎ，より高い分類精度と，視覚的にも自然な結果が得られると期待される．

FPGA を用いたアクセラレーションの導入も，有益である．現時点での GPU による動作でもフレームレートの面では十分であるものの，大型のデスクトップ PC を用いるため携行性に乏しく，消費電力も大きい．また，カメラの解像度を向上させると，より多くの情報を利用できることから分類精度の向上に寄与する可能性があるが，この場合，GPU では十分なパフォーマンスが維持できなくなると考えられる．3.6.5 項では，FPGA にフルパイプライン実装した畳み込みニューラルネットワークが，1920×1080 (フル HD) で 60 fps の動作速度を達成できることを示している．今回の評価で最良の精度を達成した Flip L4+SD 構成は，パラメータ数が小さく抑えられているため，同様のパイプラインによる実装が可能であると考えられる．この場合，再帰構造をパイプラインでどのように扱うかが問題であるものの，ストライドによりダウンサンプリングが行われることから，スケールの異なる複数の特徴マップを 1 つのストリームにインタリーブするなどの工夫により，実現できると見込まれる．また，特にクラス分類問題において資源量削減に大きな効果を持つ，2 値化ニューラルネットワーク [7, 8, 10] 等の量子化技術を導入することで，より大きなネットワークへも対応できる．

最後に，本システムは医療現場への導入を目指すものであるから，ロボットと組み合わせた実践的な評価実験により，安全性と有効性を検証しなければならない．すでに，腹腔鏡を制御するためのロボットと組み合わせた動物実験を実施しており，今後もシステムの実用化に向けた各種実験を行っていく予定である．

第5章

画像ベースのリアルタイム振動検出システムの実装と評価

本章では，マイクロサージェリーにおける外科医の補助を目的とした，画像ベースのリアルタイム振動検出システムの設計及び FPGA への実装を示し，その性能を評価する．

5.1 背景と目的

コンピュータを用いた情報処理と，センサやアクチュエータによる現実世界とのインタラクションを密結合させるサイバーフィジカルシステム (Cyber-Physical System, CPS) は，多くの分野において注目を集めており，その一つが医用工学 (Medical Engineering) である．例えば，マイクロスコープを用いることにより，肉眼では難しい精緻な手術を可能とするマイクロサージェリーでは，外科医の手に生じる不随意の細かな震えである，生理的振戦が大きな問題となる．そこで，振戦を検知し，逆位相の振動を器具に加えて抑制するマイクロサージェリー支援システムの構築が考えられる．

このような振戦抑制システムの実現にあたっては，大きく分けて二つの技術的な困難がある．一つ目は，マイクロサージェリーに用いる手術器具の上に加速度センサ等を取り付けることは，繊細な手術の妨げとなるため望ましくないということである．特に，制振すべき箇所である器具先端部にセンサを取り付けることは難しい．この問題に対する有望なアプローチとして，入力される動画像から，リアルタイム画像処理により振動成分を検出するという方法が挙げられる．カメラから得られた画像をディスプレイに表示しながら行うヘッドアップ手術はすでに一般的となっているため，カメラの導入は手術の妨げとはならない．しかしながら，手術器具の先端をトラッキングするような単純な画像処理では頑健性に欠ける．二つ目の問題は，外科医の自発的な運動を妨げることのないよう，検出した振動成分に対するフィルタリングの仕組みが必要であることである．逆位相の振動による振戦抑制のためには，フィルタリングにより発生する位相遅れを最小限に抑えなければならず，位相回転が生じる通常のバンドパスフィルタは適さない．

本研究では，この二つの問題に対処するための手法を提案する．前者に対しては，密なオプティカルフロー (Dense Optical Flow) を用いて動画像から振動成分を取得するアプローチを採る．特徴点トラッキングのような手法と異なり，このアプローチは特定の画素の情報に強く依存せず，画像内の広範囲におけるオプティカルフロー場の統計情報を利用するため，頑健な結果が得られる．オプティカルフローの算出には多量の計算が必要で

あるため，Lucas-Kanade (LK) 法 [55] を簡略化して利用する．また後者に対しては，適応型バンドパスフィルタの一種である Band-Limited Multiple Fourier Linear Combiner (BMFLC) [56] を導入する．BMFLC は入力信号を，事前に定義された周波数帯域内にある複数の正弦波を合成したものとして再構成するものであり，位相遅れのないバンドパスフィルタとして振る舞う．さらに，組み込みのフレームワークでリアルタイム処理を実現するために，オプティカルフロー算出と BMFLC によるフィルタリングを FPGA 上に実装する．深いパイプライン構造を持つハードウェア設計により，画像データの取得と処理を同時に行い，高いレベルの実行効率を実現する．

5.2　関連研究

本節では，振戦検出及び抑制の実現を狙った関連研究について，画像情報を用いるものと用いないものに大別して，簡単にまとめる．

5.2.1　画像情報を用いない振戦検出・抑制

振戦の抑制については，様々な観点に基づいて多くの研究がなされてきた．Riviere らは，生理的振戦の推定及びキャンセリングのため，Weighted Frequency Fourier Linear Combiner (WFLC) と呼ばれる適応型アルゴリズムを提案した [57]. WFLC は，フーリエ級数モデルの振幅や位相，周波数を適応させることで，準周期信号を推定する．システムはデスクトップ PC に実装され，振戦抑制に必要なリアルタイム性を達成した．Veluvolu らは，WFLC のコンセプトを拡張し，複数の周波数成分を含む入力信号も効果的に扱えるようにした Band-Limited Multiple Fourier Linear Combiner (BMFLC) [56] を提案した．著者らの振戦抑制システムでは，BMFLC をリアルタイム OS 上のソフトウェアとして実行し，加速度センサから得たデータを処理している．

Rocon らは，WFLC に基づいた振戦抑制システムを，リハビリテーション用のロボット外骨格に実装した [58]．ここでは，振動成分はジャイロスコープを用いて取得している．また As'arry らは，粒子群最適化 (Particle Swarm Optimization) と差分進化 (Differential Evolution) を用いて，振戦抑制のための作用力制御を行う仕組みを設計し，シミュレーションにより有効性を評価した [59]．Sajith らは，振動成分の検出のために触覚デバイスを用いた振戦分析システムを実装した [60]．さらに Case らは，振戦抑制システムのサイズや重量を削減するために，磁性流体ダンパーを用いることを提案した [61]．このシステムでは，角度位置を計測するためにオプティカルエンコーダを用いている．

5.2.2　画像情報を用いる振戦検出・抑制

Soran らは，サポートベクターマシン (Support Vector Machine, SVM) に基づくモーションフィルタリング技術を用いた振戦検出システムを開発した [62]. このシステムの処理は，肌領域の検出，Lucas-Kanade (LK) オプティカルフローによる特徴量抽出，周波数領域における SVM を用いた振動情報検出の 3 段階に分けられる．周波数成分に着目したアプローチにより，高い認識精度を達成したものの，振戦抑制に不可欠な時間領域にお

ける情報は検出できない．一方 Cuppens らは，Horn-Schunck オプティカルフロー [63] に基づいた画像ベースのてんかん検出アルゴリズムを提案した [64] が，計算コストが大きいため，振戦検出への適用は難しい．Uhrikova らも画像ベースの振戦分析システムを実装した [65]. しかしながら，この手法は振動周波数の測定を目的とするものであるため，振戦検出に直接適用することはできない．

5.3　アルゴリズム

本節では，本研究のリアルタイム振戦検出システムにて用いられる主要な 2 つのアルゴリズム，Lucas-Kanade (LK) オプティカルフローと Band-Limited Multiple Fourier Linear Combiner (BMFLC) についてその概要を述べる．

5.3.1　オプティカルフローと Lucas-Kanade 法

オプティカルフローは，動画像における隣接フレーム間の物体の動きを 2 次元ベクトル場として表現したものである．時刻 t における座標 $(x,\ y)$ の画素 $I(x,\ y,\ t)$ が，時刻 $t+\Delta t$ において $(x+\Delta x,\ y+\Delta y)$ に移動したとし，移動後にも画素値 (輝度値) は変化しないと仮定すると，以下の式が得られる．

$$I(x,\ y,\ t) = I(x+\Delta x,\ y+\Delta y,\ t+\Delta t) \tag{5.1}$$

ここで，画像が微分可能 (変化が滑らか) であり，また画素の移動量は小さいと仮定して，右辺を 1 次の項までテイラー展開すると

$$I(x+\Delta x,\ y+\Delta y,\ t+\Delta t) \simeq I(x,\ y,\ t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t \tag{5.2}$$

式 (5.2) を式 (5.1) に代入し，両辺を Δt で割って整理すると，

$$\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t} + \frac{\partial I}{\partial y}\frac{\Delta y}{\Delta t} + \frac{\partial I}{\partial t} \simeq 0$$

となる．簡単のため文字を置き換えると，以下の**オプティカルフローの拘束方程式**

$$I_x u + I_y v + I_t = 0 \tag{5.3}$$

が近似的に成り立つ．ここで $I_x = \partial I/\partial x$ および $I_y = \partial I/\partial y$ は画素値の水平および垂直方向の勾配を表しており，$I_t = \partial I/\partial t$ は時間微分 (フレーム間の画素値の変化) を表している．また，$(u,\ v) = (\Delta x/\Delta t,\ \Delta y/\Delta t)$ が $(x,\ y)$ におけるオプティカルフローである．

なおこの式は，以下のようにも導ける．あるフレームにおける座標 $(x',\ y')$ の画素値 $I_1(x',\ y')$ を，$(x,\ y)$ における勾配 I_x，I_y を用いて平面で近似すると，

$$I_1(x',\ y') = I_x(x'-x) + I_y(y'-y) + I_1(x,\ y)$$

となる．続いて，次のフレームで，画像全体が $(u,\ v)$ だけ平行移動したとすると，このフレームにおける画素値 $I_2(x',\ y')$ は，

$$I_2(x',\ y') = I_x((x'-u)-x) + I_y((y'-v)-y) + I_1(x,\ y)$$

と表される．したがって，

$$I_2(x',\ y') - I_1(x',\ y') = -I_x u - I_y v$$

となり，$I_t = I_2(x',\ y') - I_1(x',\ y')$ とおけば，式 (5.3) が導かれる．

式 (5.3) は 2 つの未知数 u と v を含むため，この式単独では解を一意に定めることはできない．そこで，オプティカルフローを推定するためにいくつかの手法が提案されているが，今回の研究では Lucas-Kanade (LK) 法 [55] を用いる．この手法では，以下の追加条件を仮定する．

- 対象物体上の各画素の動きは 1 画素未満と小さい
- 小さな局所領域内においては，物体の動きは一様である

これらの条件下においては，式 (5.3) が局所領域内の全ての画素で成立する．したがって，以下のように最小二乗法を用いることで，オプティカルフロー $(u,\ v)$ を近似できる．

$$\begin{pmatrix} u \\ v \end{pmatrix} = -\begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1} \begin{pmatrix} E \\ F \end{pmatrix} \tag{5.4}$$

$$\begin{aligned} A &= \sum w_i I_{x,i}^2,\ B = C = \sum w_i I_{x,i} I_{y,i}, \\ D &= \sum w_i I_{y,i}^2,\ E = \sum w_i I_{x,i} I_{t,i},\ F = \sum w_i I_{y,i} I_{t,i} \end{aligned}$$

ただし，i は局所領域内における画素のインデックスであり，w_i は，中央部に近い画素を重視するための重み係数である．今回の実装では，局所領域として 3×3 の正方形を用い，w_i にはガウシアンカーネルを用いることとする．

5.3.2　BMFLC

振戦抑制のためには，自発的な動作を妨げないためのバンドパスフィルタが必要であることは 5.1 節ですでに述べた．しかし，一般的なバンドパスフィルタでは位相の回転が生じるため，振動抑制に効果的な逆位相信号の生成が難しい．そこでよく用いられるアプローチが，適応型アルゴリズムの採用である．入力信号を何らかの関数で近似できれば，次の時間における信号の値を推定できるため，位相遅れを回避できる．信号は時々刻々と変化するが，内部パラメータを随時更新することにより，変化に追従し続ける．

Band-Limited Multiple Fourier Linear Combiner (BMFLC) [56] はこうした適応型推定フィルタの一種であり，事前定義された周波数範囲内に限定されたフーリエ級数モデルに基づいている．図 5.1 に示すように，注目する周波数範囲 $[f_0,\ f_1]$ を，同一の周波数幅を持

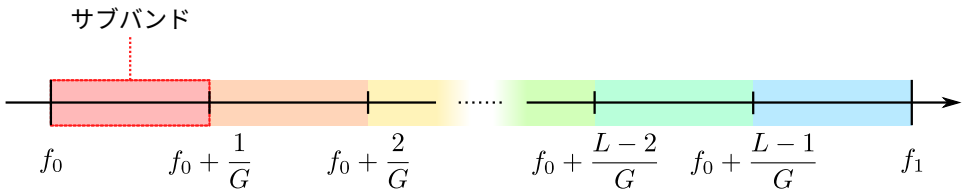


図 5.1: G に基づく周波数範囲の分割

つ L 個のサブバンドに分割する．それぞれのサブバンドは，$L = (f_1 - f_0)G$，$f_r = f_0 + \frac{r}{G}$ とすると $[f_r,\ f_{r+1}]$ $(0 \leq r < L)$ と表される．

BMFLC の出力は，入力信号 s_k の推定値 $\hat{s_k}$ であり，以下のように求められる．

$$\begin{aligned} \hat{s_k} &= \sum_{r=0}^{L-1} \{w_{r,k} \sin(2\pi f_r t_k) + w_{r+L,k} \cos(2\pi f_r t_k)\} \\ &= \boldsymbol{w}_k^T \boldsymbol{x}_k \end{aligned} \tag{5.5}$$

ただし，k はサンプルのインデックス，t_k [sec] は k における時刻, $\boldsymbol{w}_k = [w_{1,k},\ \cdots,\ w_{2L,k}]^T$ は適応重みベクトルである．また，$\boldsymbol{x}_k = [x_{1,k},\ \cdots,\ x_{2L,k}]^T$ は以下の式により与えられる参照入力ベクトルである．

$$x_{r,k} = \begin{cases} \sin(2\pi f_r t_k) & (0 \leq r \leq L-1) \\ \cos(2\pi f_{r-L} t_k) & (L \leq r \leq 2L-1) \end{cases} \tag{5.6}$$

重みベクトル $\boldsymbol{w}_k$ は，式 (5.5) におけるフーリエ係数に対応しており，Least Mean Square (LMS) アルゴリズム [66] により，以下のように更新される．

$$\varepsilon_k = s_k - \hat{s_k} = s_k - \boldsymbol{w}_k^T \boldsymbol{x}_k \tag{5.7}$$

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + 2\mu\varepsilon_k \boldsymbol{x}_k \tag{5.8}$$

ここで μ は適応ゲインパラメータであり，収束の速度と安定性のバランスを取る役割を担っている．参照入力ベクトル $\boldsymbol{x}_k$ は，式 (5.6) に示されているように帯域制限された正弦波信号から成るため，BMFLC は位相遅れのないバンドパスフィルタとして振る舞う．表 5.2 に，BMFLC の処理の流れを総括して示す．

5.4　設計

本節では，5.3 節で述べたアルゴリズムに基づいた，本研究における，画像情報からの振動成分検出システムの設計について，モジュールの実装にも触れながら述べる．

5.4.1　設計指針

画像ベースの振戦抑制のために，システムは入力されるカメラの動画像からリアルタイムかつ低レイテンシに振動成分を検出しなければならない．このことから，本研究ではオ

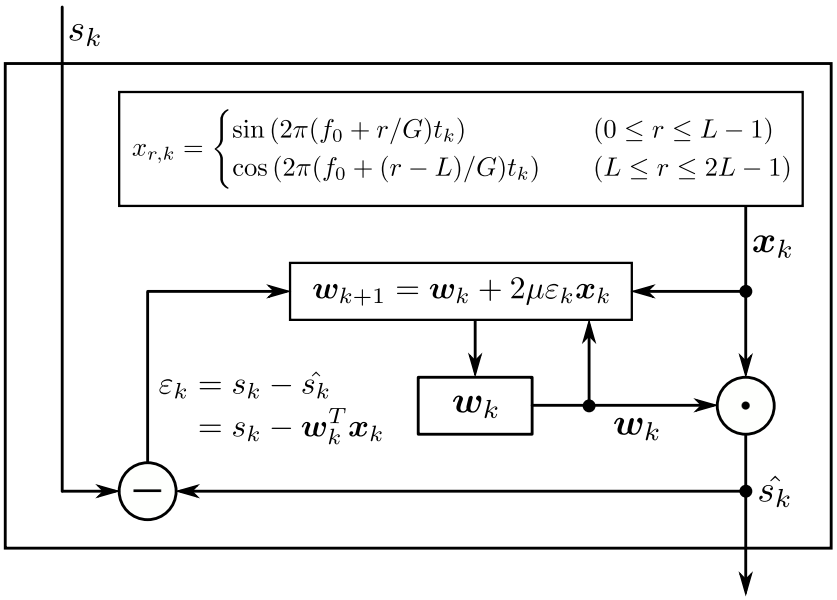


図 5.2: BMFLC の処理の流れ

プティカルフローの計算処理を，完全パイプライン化されたストリーム処理モジュールとして実装することとした．このモジュールは，カメラの解像度やフレームレートの影響を受ける．具体的には，ピクセルクロックにより定められるタイミング制約を満たさなければならないし，オプティカルフローの計算に必要なフレームバッファが利用可能なメモリ資源量に収まらなければならない．

解像度に関しては， 高解像度では大きなフレームバッファが必要となり，大量のメモリ資源を消費してしまうことから，VGA (640×480) を用いることとする．VGA のカメラが 90° の水平視野角を持ち，手から 50 cm 離れて設置されていると仮定すると，1 画素は手の位置において約 1.56 mm に相当する．また，LK オプティカルフローがサブピクセルの精度を持つことから，1.56 mm 未満の振幅の振戦も検出可能である．したがって，VGA 解像度は実用的な設定であると言える．

[67] で示された振戦の特性に基づき，本研究では周波数範囲が 8 から 12 Hz の振動成分の検出を目指す．12 Hz の振動を検出するためには，フレームレートは最低でも 24 fps (frames per second) が必要であり，より高いほうが望ましい．動画像のフレームレートとしては 30 あるいは 60 fps が一般的であり，フレームレートはフレームバッファの大きさに影響しないことから，最大動作周波数の許す限り高く設定できる．そのため，今回はフレームレートとして 60 fps をターゲットに定めた．

また，効果的な振戦抑制を達成するためには，BMFLC を用いたバンドパスフィルタのレイテンシを小さく抑えなければならない．正弦波信号 $\sin\theta$ を，逆位相の正弦波信号 $-\sin\theta$ でキャンセルする場合を考える．逆位相信号に d rad の位相遅れがあると仮定す

ると，キャンセル後の信号は，以下のように表される．

$$\begin{aligned}\sin\theta - \sin(\theta - d) &= \sin\theta - \sin\theta\cos d + \cos\theta\sin d \\ &= (1-\cos d)\sin\theta + \sin d\cos\theta \\ &= \sqrt{(1-\cos d)^2 + \sin^2 d}\sin(\theta+\alpha) \\ &= \sqrt{2(1-\cos d)}\sin(\theta+\alpha)\end{aligned}$$

信号の周波数を f Hz，逆位相信号のレイテンシを Δt sec とすると $d = 2\pi f\Delta t$ であるから，キャンセルにより，振幅は $\sqrt{2(1-\cos(2\pi f\Delta t))}$ 倍に軽減されることになる．ここから，12 Hz の振動の振幅を 90% 低下させるために許されるレイテンシは 1.3 msec 程度であり，99% の抑制の場合は 130 μsec となることが分かる．

5.4.2　設計概要

5.4.1 で定めた要件に基づいて，システムの設計を行った．図 5.3 は，提案システムの構成を大まかに示したものである．システムは，*of* モジュール，*of2bmflc* モジュール，そして *bmflc* モジュールから構成されている．まず，カメラから入力される 8 ビットのグレースケール画像は，*of* モジュールにストリームとして入力される．つまり，1 クロックにつき 1 画素が，ラスタスキャンの要領で入力される．*of* モジュールは LK オプティカルフロー $(u,\ v)$ を各画素について計算するもので，完全にパイプライン化されている．*of* モジュールの出力は *of2bmflc* モジュールに与えられ，ロバストな平均オプティカルフロー $(\bar{u},\ \bar{v})$ がフレームごとに生成される．そして最終的に *bmflc* モジュールが，事前に定義された周波数範囲内の振動成分 $(\Delta x,\ \Delta y)$ を推定し，出力する．

of と *of2bmflc* モジュールはピクセルクロックと同期して動作する一方，*bmflc* モジュールは，5.5 節で説明するように，より高速な別のクロックにより駆動される．クロックドメインをまたいだデータの受け渡しには，シンプルなハンドシェイクに基づくシンクロナイザを用いた．$\bar{u}$ と $\bar{v}$ は 1 フレームに 1 回しか生成されないため，この手法で十分な通信速度が得られる．

5.4.3　*of* モジュール

of モジュールは，図 5.4 及び 5.5 に示すような深いパイプライン構造により，入力画像から各画素の LK オプティカルフロー $(u,\ v)$ を計算する．これは，[68] から着想を得たものである．

図 5.4 はモジュールの前段部であり，オプティカルフローの計算に必要な水平方向の勾配 I_x，垂直方向の勾配 I_y，そして時間方向の変化 I_t を計算する役割を担う．まず，入力されたフレーム k の画素 p_k の周辺 3×3 画素を，FIFO とシフトレジスタを組み合わせた移動ウインドウにより得る．この小画像に，水平及び垂直方向のソーベルフィルタを適用することで，I_x および I_y が得られる．なお，ソーベルフィルタのカーネルについては図 5.4 に示すとおりであり，絶対値として 0，1 もしくは 2 しか含まないため，シフトと加減算器のみで実装でき，乗算器を必要としない．一方 I_t については，FIFO として構

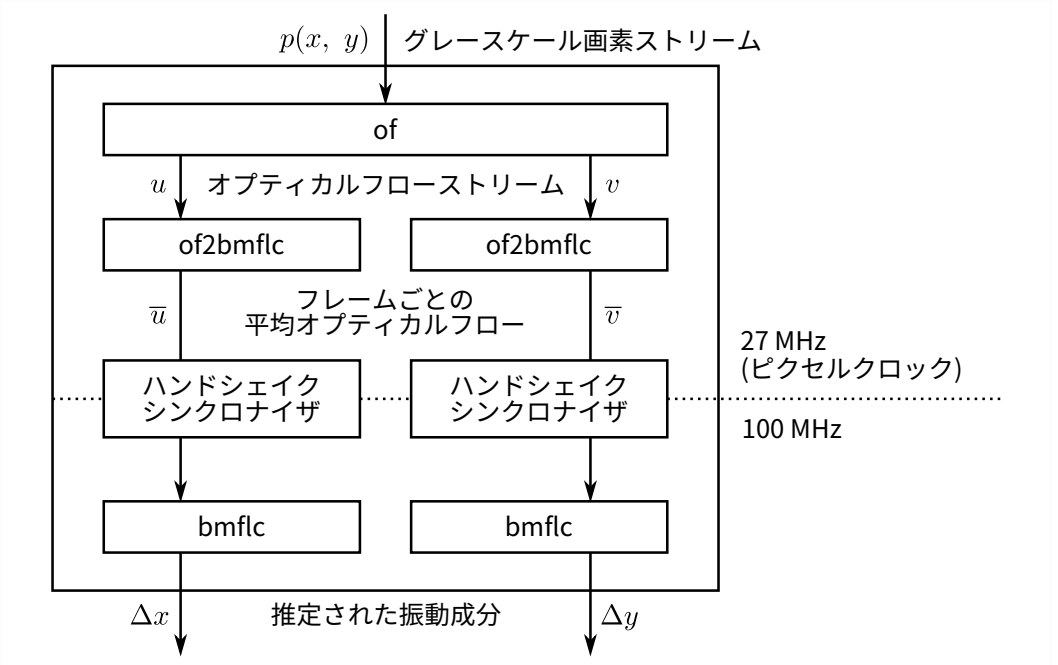


図 5.3: 提案システムの構成図

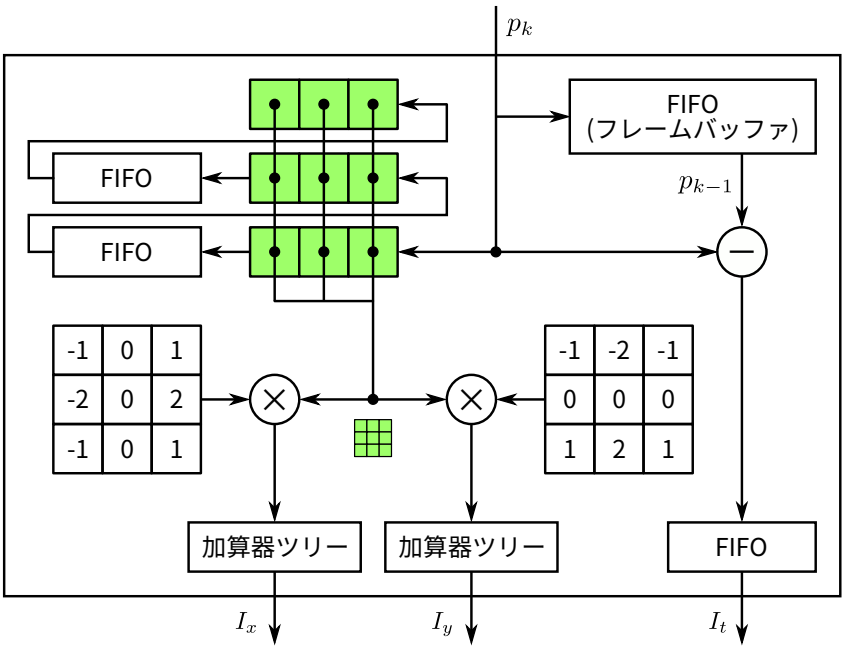


図 5.4: *of* モジュールの前段部の構成

築されたフレームバッファにより，前のフレームにおける画素値 p_{k-1} を取得し，p_k との減算を行うことで得られる．I_x および I_y とタイミングを合わせるためのバッファが挿入されていることに注意されたい．

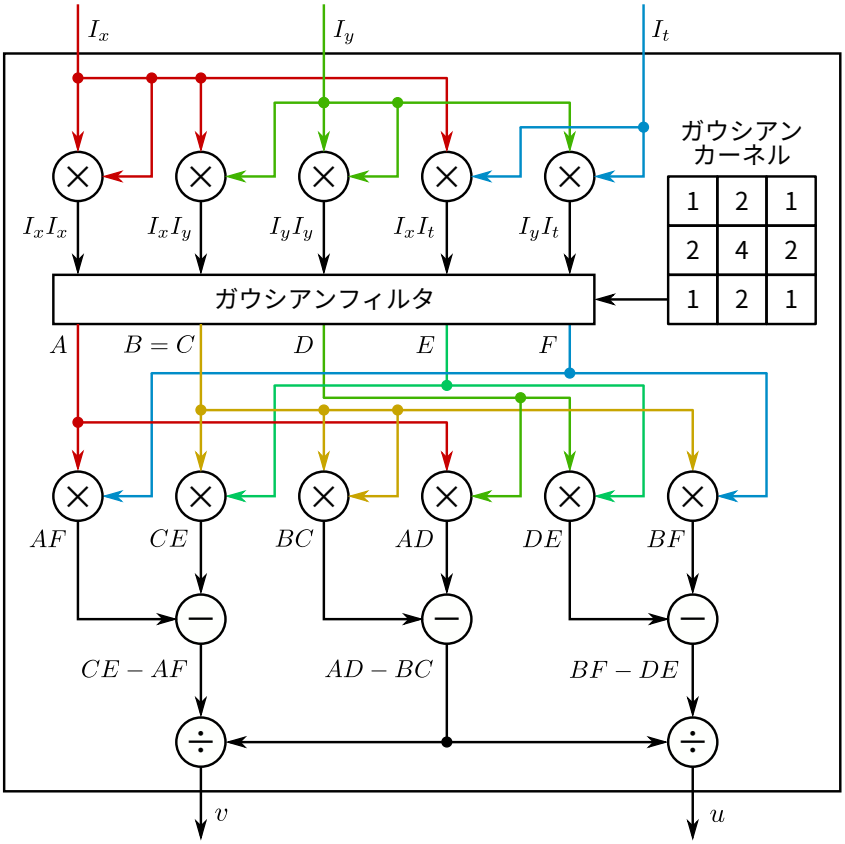


図 5.5: *of* モジュールの後段部の構成

図 5.5 に示すモジュール後段部では，前段部で求めた I_x, I_y, I_t を用いて，5.3.1 節の式 (5.4) を以下のように解くことで，各画素の LK オプティカルフロー $(u,\ v)$ を求める．

$$\begin{aligned}\begin{pmatrix} u \\ v \end{pmatrix} &= -\begin{pmatrix} A & C \\ B & D \end{pmatrix}^{-1} \begin{pmatrix} E \\ F \end{pmatrix} \\ &= \frac{1}{AD - BC} \begin{pmatrix} BF - DE \\ CE - AF \end{pmatrix}\end{aligned} \tag{5.9}$$

まず，I_x^2, I_xI_y, I_y^2, I_xI_t および I_yI_t のストリームを得るために，5 つの乗算が並列に行われる．続いて，図 5.4 のソーベルフィルタと同様の仕組みにより，3×3 ガウシアンカーネルの畳み込みを，各ストリームに対して行う．カーネルの値は図 5.5 内に示すとおりであり，今回の実装では全て 2 の冪乗となっているため，ソーベルフィルタと同様，乗算器は必要でない．ガウシアンフィルタ適用後，式 (5.9) に示した逆行列の乗算により，オプティカルフロー $(u,\ v)$ を求める．ここで，図 5.5 の大部分の演算は整数演算により行えるが，最終段のパイプライン除算器についてのみ，固定小数点演算を用いている点に

注意されたい．そのため，オプティカルフローの最終出力 u および v は，整数部 11 ビット，小数部 n ビットの計 $(11+n)$ ビットで表現される．整数部は計算式から定まるダイナミックレンジに基づいて決定されており，n はパラメータで指定可能である．

一般的には，ナイーブな LK 法により得られるオプティカルフローの精度は限られたものとなる．これは，5.3.1 節において示した，アルゴリズムが前提とする仮定が必ずしも成立しないためである．この問題に対処するため，1 画素を超える移動量に対応するための解像度ピラミッドの利用 [69] 等，多くの洗練された改良手法が提案されている．しかし，本研究の設計では，密なオプティカルフローの平均をフレームごとに取ることで振動成分を抽出する仕組みを採ることで，頑健性が確保される．これを考慮し，ハードウェアのパイプライン構造をシンプルに留め，さらにレイテンシの増大を回避するため，ナイーブな LK 法によるアプローチを採用することとした．

5.4.4　*of2bmflc* モジュール

of2bmflc モジュールは，フレームごとにオプティカルフローのロバストな平均値を計算する．本モジュールは u 及び v のそれぞれに対して別に用意されており，その構造は，図 5.6 に示されるとおりである．このモジュールは入力されたオプティカルフローの値を積算し，フレーム内のオプティカルフロー値の総和を求める．ただし，ノイズの影響を緩和するため，絶対値がしきい値 $t>0$ を上回るオプティカルフローのみを利用するようにしている．現在 $t=1.0$ に設定している．また，さらなるノイズ軽減を図るため，オプティカルフローの合計値が正である場合には正のオプティカルフローの平均を，負である場合には負のオプティカルフローの平均を出力する符号フィルタリングの仕組みも導入している．これは，全体としてある方向のフローが強い場合は，逆方向のフローはノイズとみなして取り除くという発想に基づくものである．この仕組みはパラメータにより有効無効を任意に切り替えることができ，無効である場合には，出力は単に，絶対値が t を超える全てのフローの平均値となる．

of モジュールとは異なり，本モジュールに搭載された，平均値 $\bar{u}$ あるいは $\bar{v}$ を求めるための除算器はパイプライン化されていない．これは，除算を行う必要があるのは 1 フレームごとに 1 回だけであるためである．この除算器は，同一の比較器と減算器を繰り返し使用することで引き戻し法で商を求めており，*of* モジュールのパイプライン除算器と比較してはるかに少ない資源しか必要としない．

5.4.5　*bmflc* モジュール

$\bar{u}$ と $\bar{v}$ のそれぞれに対して用意され，BMFLC によるバンドパスフィルタを適用する *bmflc* モジュールは，4 状態のステートマシンにより制御されている．状態間で受け渡されるデータと，各状態で行われる処理を図示したものが図 5.7 である．本モジュールは，各状態での処理はパイプラインで行われるものの，モジュール全体での完全パイプライン構造にはなっていない．これは，$\bar{u}$ と $\bar{v}$ は 1 フレームに 1 回しか与えられず，完全パイプラインで構築する必要がないためである．こうすることで，状態間で共通する演算では演算器を共有でき，資源使用量の削減につながる．以下，*bmflc* モジュールの入力信号 (フレーム k における $\bar{u}$ または $\bar{v}$) を s_k と表記する．

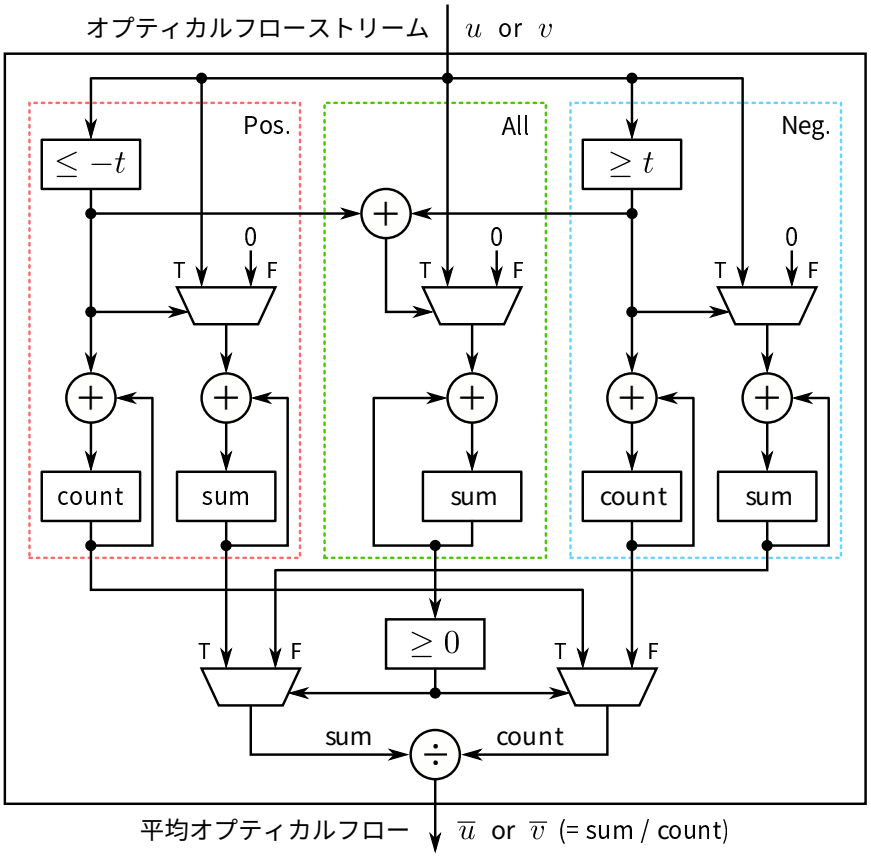


図 5.6: *of2bmflc* モジュールの構成

State 0 においては，入力された位相 θ に対応する正弦関数 $\sin\theta$ の値を出力する *sin_calc* モジュールを用いて，次フレームの時刻 t_{k+1} に対応する，式 (5.6) で示した参照入力ベクトル $\boldsymbol{x}_{k+1}$ を計算する．このモジュールは，内部のルックアップテーブル (LUT) に，$\theta \in [0, \pi/2]$ に対応する正弦関数 $\sin\theta$ の値計 1024 エントリを保持している．なお，この LUT は Block RAM を用いた ROM として実装されている．位相 θ $(0 \leq \theta < 2\pi)$ が 12 ビットの整数として与えられると，下位の 10 ビットをテーブルのインデックス計算に，残りの 2 ビットを象限の選択に利用して，$\sin\theta$ を出力する．$\cos\theta = \sin(\theta + \pi/2)$ の関係を利用することで，$\boldsymbol{x}_{k+1}$ 全体をこのモジュール 1 つで用意できる．

State 0 における参照入力ベクトルの準備が完了すると，*bmflc* モジュールは State 1 に遷移し，*of2bmflc* からフレームごとの平均オプティカルフロー s_k が送信されるまで待機する．その後 State 2 において，前回のサイクルで用意された参照入力ベクトル $\boldsymbol{x}_k$ に $2\mu\varepsilon_k = 2\mu(s_k - \hat{s}_k)$ が乗じられ，式 (5.7) 及び (5.8) で示したように，その積が現在の重みベクトル $\boldsymbol{w}_k$ に加算されることで，重みベクトルが $\boldsymbol{w}_{k+1}$ に更新される．最終的に State 3 において，式 (5.5) に従い，$\boldsymbol{x}_{k+1}$ と $\boldsymbol{w}_{k+1}$ の内積が計算され，推定された信号値 $\hat{s}_{k+1}$ として出力されることになる．なお，State 2 および 3 の全演算は，$(11 + n)$ ビットの固定小数点演算である．

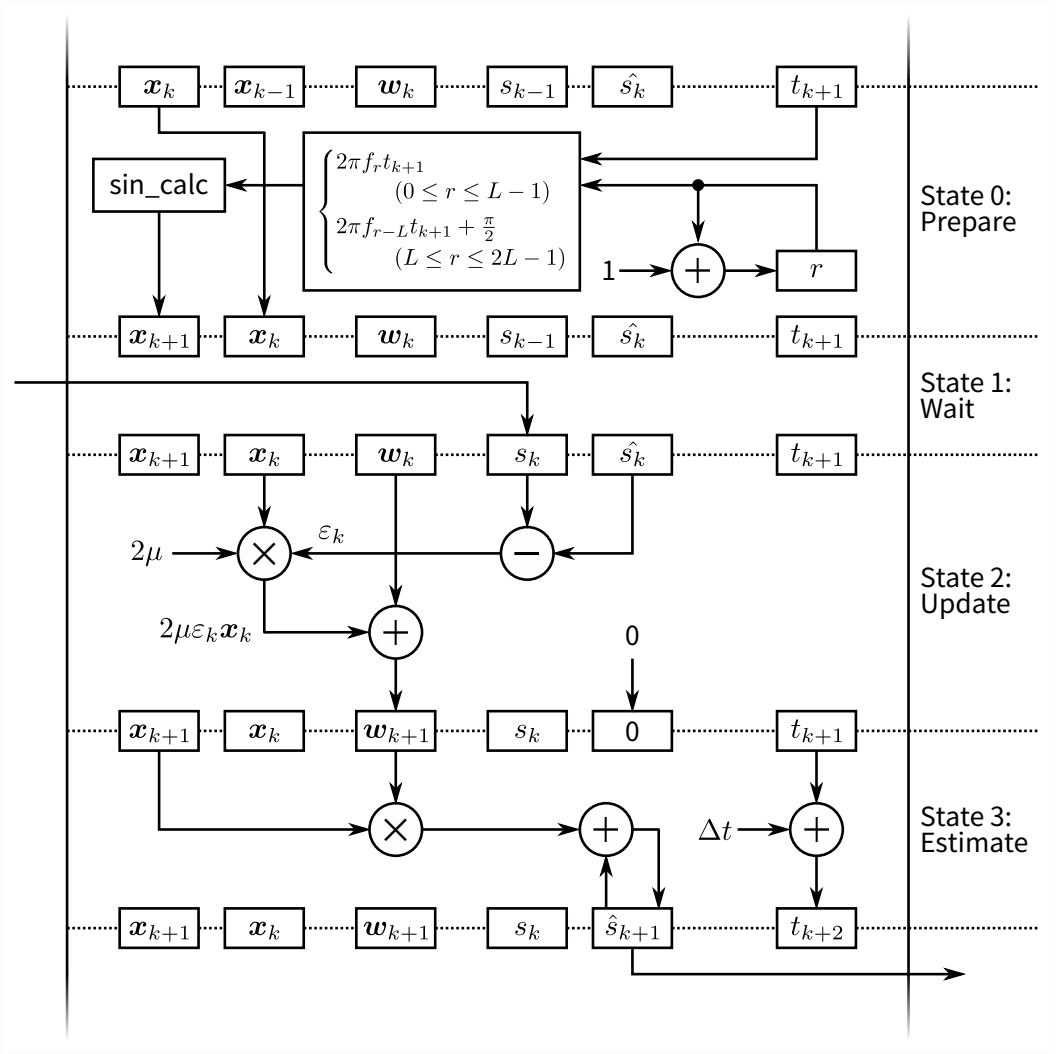


図 5.7: *bmflc* モジュールの処理の流れ

今回の実装では，設計パラメータは以下のように設定している．

$$\mu = 2^{-7},\quad G = 4,\quad f_0 = 8,\quad f_1 = 12$$

つまり，ターゲットとする周波数範囲は 8 から 12 Hz であり，調波の数 L は $L = (12 - 8) \times 4 = 16$ である．周波数範囲の設定は，生理的振戦の特性 [67] を考慮して行った．

5.5　評価と考察

本節では，5.4 節の設計に基づいて構築された本研究のリアルタイム振戦検出システムの性能評価を行う．

5.5.1　ハードウェア実装

提案システムは，SystemVerilog を用いたレジスタ転送レベル (Register Transfer Level, RTL) 記述で FPGA に実装した．FPGA にパイプライン実装することでレイテンシが抑えられるのみならず，その再構成可能な性質により，ASIC (Application-Specific Integrated Circuit) と比較して，カメラの I/O インタフェースや解像度が変更された場合など，多様な環境にシステムを適応させやすくなる．加えて，振戦抑制を実現するには，本研究の振戦検出システムを手術器具に取り付けたアクチュエータシステムと連携させなければならないが，FPGA を用いることで，このような拡張も容易に行える．

論理合成および実装には Xilinx Vivado 2018.3 を用い，ターゲットとする FPGA は，KC705 評価ボード上に搭載された Xilinx Kintex-7 XC7K325T FPGA である．評価にあたっては，OmniVision Technologies 社製の OV9620 CMOS カメラを入力デバイスとする．このカメラは，VGA 解像度 (640×480) の映像を 60 fps で出力する．

of および *of2bmflc* モジュールについては，図 5.3 にもあるように，カメラデバイスのピクセルクロックに同期した 27 MHz のクロックを用いる．この周波数は近年の FPGA 回路としては低いが，本システムはパイプライン構造をとるため，VGA 解像度の画像で 60 fps を達成するために十分な周波数となる．また，ピクセルクロックに同期した設計は，ハードウェアによる画像処理において電力効率で有利であることが示されている [70]. 一方 *bmflc* モジュールについては，100 MHz のクロックで駆動される．これは KC705 評価ボードで提供されるシステムクロック周波数の半分である．*bmflc* モジュールはフレームごとに与えられる平均オプティカルフローを処理するため，ピクセルクロックに合わせる必要はなく，高速なクロックを用いることでバンドパスフィルタのレイテンシを抑え，振戦抑制の効果を高めることを考慮した仕様である．なお，回路の最大動作周波数 $F_{\max}$ を超過しない限り，100 MHz に限らずいかなる周波数のクロックも利用可能である．

5.5.2　小数部ビット幅による精度の比較

5.4 節で言及したとおり，*of* モジュールのパイプライン除算器及び *of2bmflc* と *bmflc* モジュールでは固定小数点演算が用いられている．このアプローチの有効性を検証するため，複数の小数部ビット幅 n を用いて，固定小数点演算により発生する演算誤差を，浮動小数点演算と比較して評価した．

of，*of2bmflc* モジュールに対しては，図 5.8 に示すテスト用画像を用いて評価を行った．この図は，座標を変数とする 2 次式により画素値を変化させて生成したものである．まず，このテスト画像をフレームごとに水平方向または垂直方向に 1 画素だけシフトさせ，*of* モジュールに与えて密なオプティカルフローを計算させる．その結果を *of2bmflc* モジュールに与え，演算結果を確認する．その結果を表 5.1 にまとめる．倍精度浮動小数点数 (Float64) を用いて Python によるソフトウェア実装により生成した結果と比較すると，$n = 21$ においてはほぼ同じ結果が得られていることが分かる．この結果から，$n = 21$ ならば十分な精度を実現できると評価できる．また n が低下するにつれて，Float64 に対する誤差はコンスタントに大きくなっていくが，精度より資源使用量の低減を重視するならば，より低い設定も利用しうるであろう．

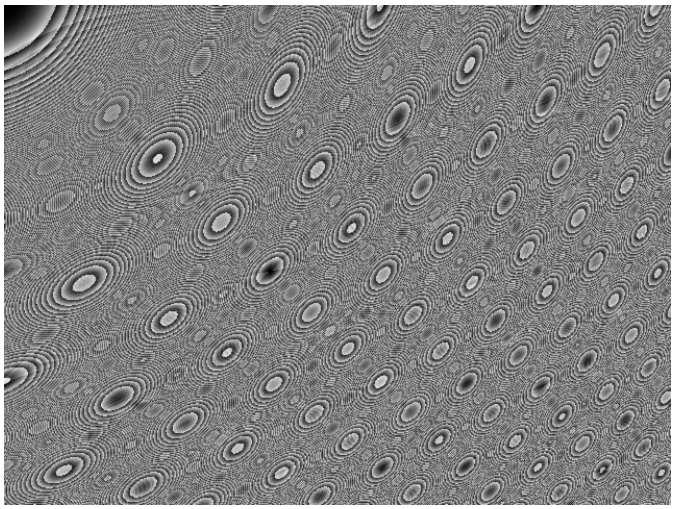


図 5.8: 固定小数点演算の評価用に生成した画像

表 5.1: 小数部ビット幅による *of2bmflc* モジュールの出力値の変化

n	9	12	15	18	21	Float64
$\bar{u}_1$	0.730469	0.731201	0.731415	0.731422	0.731426	0.731426
$\bar{v}_1$	-0.113281	-0.113770	-0.113861	-0.113869	-0.113870	-0.113870
$\bar{u}_2$	-0.035156	-0.036133	-0.036285	-0.036293	-0.036293	-0.036293
$\bar{v}_2$	0.662109	0.662842	0.663086	0.663090	0.663094	0.663094
$\bar{u}_3$	-0.765625	-0.767822	-0.767914	-0.767941	-0.767942	-0.767943
$\bar{v}_3$	0.041016	0.042480	0.042664	0.042671	0.042675	0.042675
$\bar{u}_4$	0.074219	0.074463	0.074677	0.074692	0.074694	0.074694
$\bar{v}_4$	-0.761719	-0.763184	-0.763275	-0.763287	-0.763290	-0.763290

一方 *bmflc* モジュールに対しては，2 または 10 Hz の正弦波信号を与えて評価を行った．図 5.9 に，入力信号と，それぞれの小数部ビット幅における推定出力を示す．いずれの周波数においても，$n = 21$ であれば Float64 と視覚上ほぼ同等の結果が得られていることが分かる．$n = 15$ についても良好であるが，2 Hz の信号 (図 5.9 (a)) においてわずかな誤差が見受けられる．$n = 12$ になると誤差が明確に現れ始め，$n = 9$ では信号に正しく追従できなくなっている．それぞれのビット幅における Float64 に対する平均絶対誤差 (Mean Absolute Error, MAE) を表 5.2 にまとめる．ここからも，n が増加するにつれて精度が改善していることが分かる．なお，BMFLC は適応アルゴリズムであるため，これらの誤差が経時的に蓄積することはないと考えられる．

表 5.2: 小数部ビット幅による，*bmflc* の出力の Float64 に対する平均絶対誤差

n	$f = 2$	$f = 10$
9	0.561721	0.232879
12	0.034347	0.034473
15	0.003371	0.003386
18	0.001749	0.001277
21	0.001615	0.000137

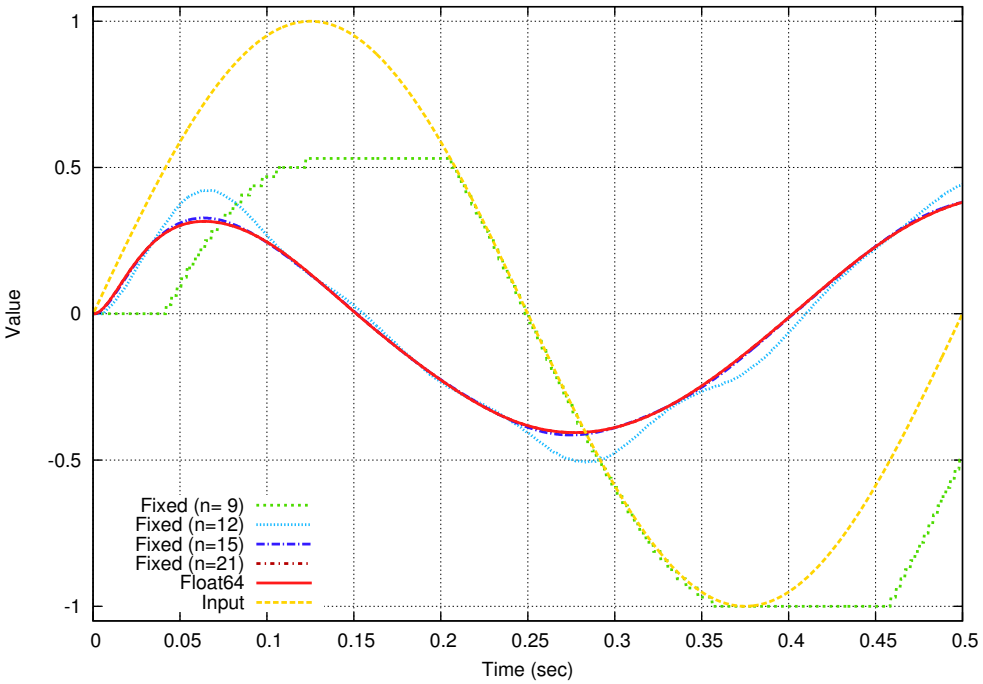
図 5.9 から，BMFLC がバンドパスフィルタとして機能していることも確認できる．(a) 2 Hz の信号は，検出対象の周波数範囲 (8 から 12 Hz) から外れているため，入力信号に対して大きく減衰している一方，(b) 10 Hz の信号は入力信号に近い振幅が得られている．また，(b) では入力と出力の位相が一致していることも分かる．このことは，BMFLC が位相遅れのないバンドパスフィルタとして振る舞うことを裏付けるものである．

結論として，小数部ビット幅を 21 ビットに設定すると，*of2bmflc* および *bmflc* いずれのモジュールにおいても Float64 に近い結果が得られた．資源使用の効率性を考慮すると，精度は十分に許容可能であると考えられる．

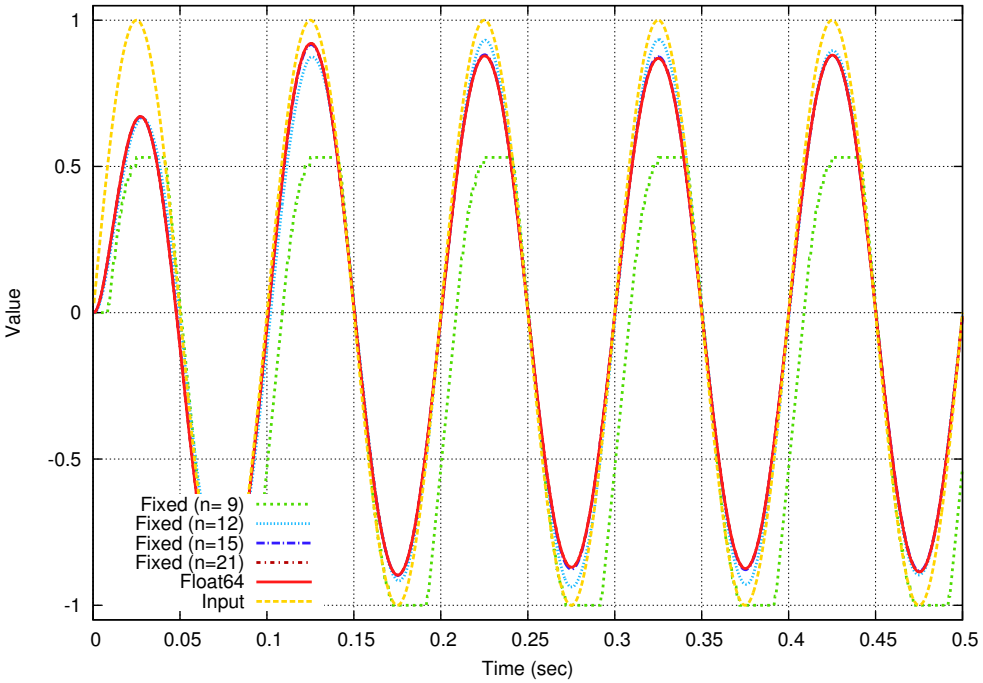
5.5.3　資源使用量

5.5.2 項での評価に基づき，小数部ビット幅として 21 ビットを採用し，Vivado を用いたターゲット FPGA への論理合成および実装を行った．その結果，設計したハードウェアは問題なく実装され，全てのタイミング制約を満たした．

表 5.3 に，FPGA の資源使用量をモジュールごとに分けてまとめる．ここから，提案システムは LUT，FF，DSP について 10% 前後の資源を消費していることが分かる．なお，LUT と FF の資源使用量合計が各モジュールの総和と一致しないが，この差はクロックドメイン間のハンドシェイクシンクロナイザによるものである．一方 Block RAM に関しては，使用率が約 39% に達している．その大部分は，5.4.3 項で説明した *of* モジュールにおけるフレームバッファに費やされている．*of* モジュールは，資源消費の大きいパイプライン除算器を含むことから，論理要素 (LUT，FF) の使用率においても支配的となって



(a) 2 Hz



(b) 10 Hz

図 5.9: 小数部ビット幅による *bmflc* の出力の変化

いる．しかし全体としては，Block RAM を除いて提案システムはコンパクトであり，組み込み環境での実装にも適することが示された．

表 5.3: FPGA の資源使用量

モジュール	LUT	FF	DSP	BRAM
of	20182	20565	24	171
of2bmflc	830	622	6	0
bmflc	3232	6954	24	1
合計	24247	28279	54	172
使用率 (%)	11.90	6.94	6.43	38.65

5.5.4　スループットとレイテンシ

今回の実装における最大動作周波数 $F_{\max}$ は，ピクセルクロックドメイン (27 MHz) では 99.5 MHz，高速クロックドメイン (100 MHz) では 113.8 MHz であった．このように，タイミング制約を十分に満たしているため，ピクセルクロックに同期したパイプラインである *of* と *of2bmflc* モジュールは，60 fps のリアルタイム性能を達成したこととなる．これらのパイプラインモジュールにおけるレイテンシは表 5.4 にまとめたとおりである．ただし，*of2bmflc* モジュールのレイテンシは，座標 (0, 0)，つまりフレームの先頭にあたる画素のオプティカルフローが入力されてから，フレーム全体の平均オプティカルフローが出力されるまでの時間とする．*of2bmflc* モジュールは，原理上，フレーム内の全画素のオプティカルフローが入力されるまで結果を出すことは不可能なため，パイプラインのレイテンシの大部分はこのモジュールによって占められている．

表 5.4: ピクセルクロックドメインのモジュールにおけるパイプラインレイテンシ

モジュール	レイテンシ (msec)	クロックサイクル数
of	0.062	1694
of2bmflc	14.218	383893
合計	14.281	385587

一方，*bmflc* モジュールでのフィルタリングに要するレイテンシは 74 クロックサイクル，つまり 0.74 μsec である．5.4.1 項で述べたように，12 Hz の振動を 99% キャンセルするために許容される最大のレイテンシは 130 μsec であることを考えると，*bmflc* モジュールにおけるレイテンシはほとんど無視できる．カメラの 1 フレームは同期領域を含めて 16.67 msec にあたることも考えると，提案システムはレイテンシの観点からも十分なリアルタイム性を達成していることが示されたと言える．

さらに，パフォーマンスの比較のため，C++ と OpenCV を用いて提案システムのソフトウェア版を実装した．このソフトウェアを，Intel Core i7 2.67 GHz CPU を搭載した

Linux デスクトップ PC で実行したところ，実行時間は 48.91 msec であり，FPGA 版が約 3 倍高速であった．FPGA が数ワットの消費電力で動作することを考慮すると，提案するアプローチが電力効率においても優れていることが示された．

5.5.5　実証実験

提案手法の正当性を検証するため，FPGA に追加のモニタリング機構を追加し，*of2bmflc* および *bmflc* モジュールの出力を外部から取得できるようにした．前者はカメラ映像から抽出された振動成分に，後者はバンドパスフィルタの適用結果に相当する．

まず，カメラの正面に機械式メトロノームを配置し，一定周波数による振り子の運動から振動成分を検出させた．BPM (Beats Per Minute) の設定として，120 と 200 の 2 つを使用する．メトロノームは，振り子が左右に振れるたびに音を発生させる器具であるため，これらの BPM は，それぞれ毎分 60 および 100 往復，つまり 1 Hz および 1.67 Hz の機械的振動に相当する．記録された結果を図 5.10 (a) から (d) に示す．120 BPM および 200 BPM における *of2bmflc* の出力に対応する図 5.10 (a) および (c) から，振り子の振動成分が適切に検出されていることが分かる．加えて，いずれの BPM に対応する周波数もバンドパスフィルタの通過帯域 (8 から 12 Hz) 外であるため，*bmflc* の出力にあたる図 5.10 (b) および (d) を見ると，検出された振動が適切に除去されていることが分かる．

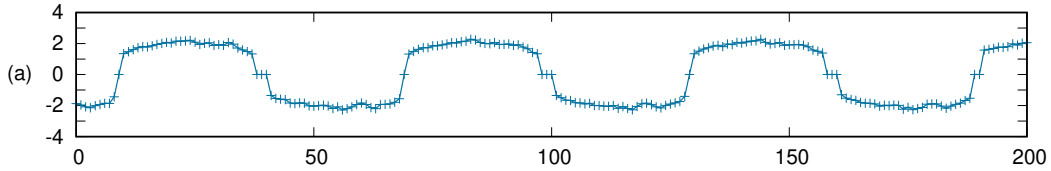
続いて，より実践的な映像における実用性を検証するため，カメラの前で手を振る実験を行った．初めは手をゆっくり動かし，その後，振戦を模擬した速い動きに切り替えた．その結果を図 5.11 (a) から (d) に示す．図 5.11 (a) および (b) から，遅い動きは *of2bmflc* により適切に検出されたが，周波数が 8 Hz を大きく下回るため，*bmflc* により正しくフィルタリングされていることが分かる．これは，提案システムを振戦抑制機構と組み合わせた場合に，ゆっくりとした自発的な手の動きを妨げないことを示している．一方，図 5.11 (c) および (d) から，振戦を模擬した速い動きは，*bmflc* で取り除かれることなく出力されていることがうかがえる．また，BMFLC が位相遅れを生じさせないことも改めて確認された．これは，振戦抑制のためのフィードバック制御において不可欠な特性である．

さらに，図 5.10 と図 5.11 の 2 実験は，一切のパラメータチューニングを行わず，同一のシステム構成で行われたことは，提案システムが，多様なシーンにおいて振動成分検出をロバストに行えることを示唆するという点において特筆すべきである．

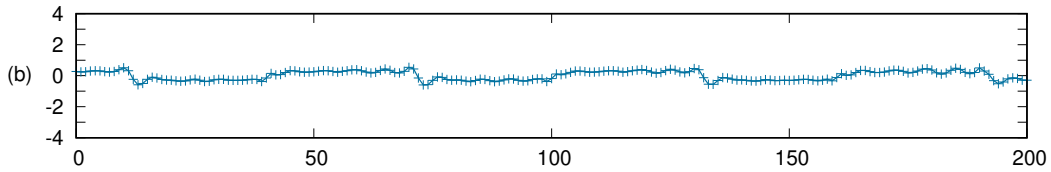
5.5.6　ニューラルネットワークを用いた検出精度の改善

手術器具の外部のオプティカルフロー成分を除去して振戦の検出精度を向上させるため，手術器具の検出システム [71] の実装を検討している．このシステムは，畳み込みニューラルネットワークのモデルの一つである ResNet [23] をベースとしており，プロトタイプのネットワークの学習にあたっては，[72] で提供されているデータセットを利用している．システムの入力画像と出力結果，正解ラベル画像の例を図 5.12 に示す．

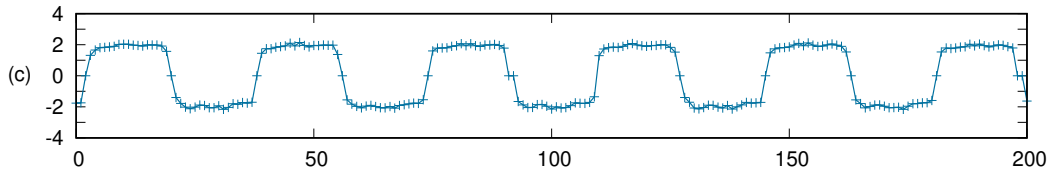
入力画像 (a) が与えられると，システムはそれぞれの画素を手術器具 (白) とその他 (黒) の 2 クラスに分類する．ネットワークの学習は，出力 (b) が正解ラベル画像 (c) に近づくように進められる．この分類により，手術器具のオプティカルフローのみに注目できるよ



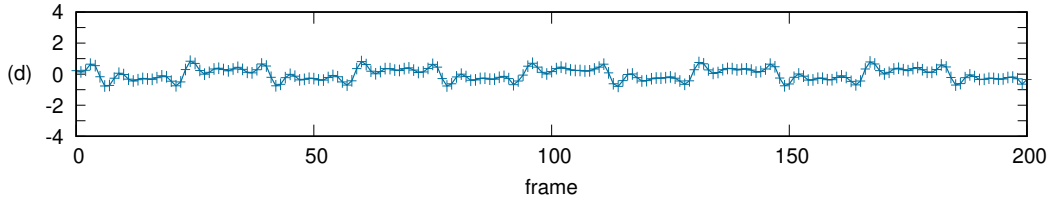
(a) 120 BPM における *of2bmflc* の出力



(b) 120 BPM における *bmflc* の出力

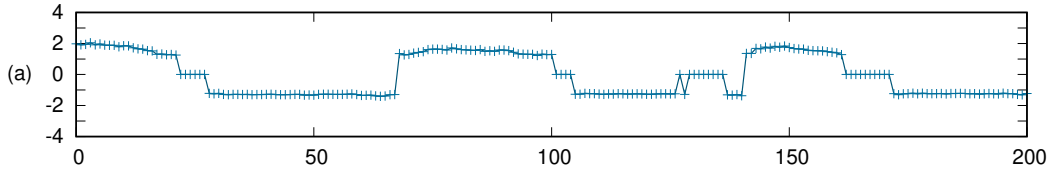


(c) 200 BPM における *of2bmflc* の出力

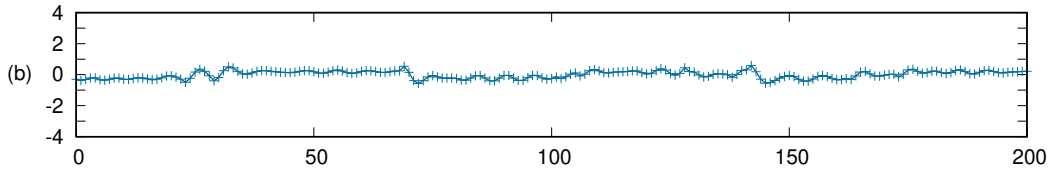


(d) 200 BPM における *bmflc* の出力

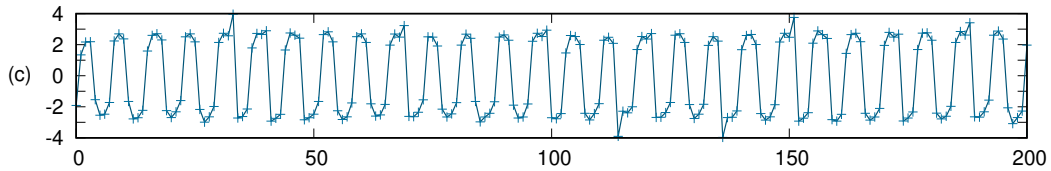
図 5.10: メトロノームの映像から抽出した振動成分



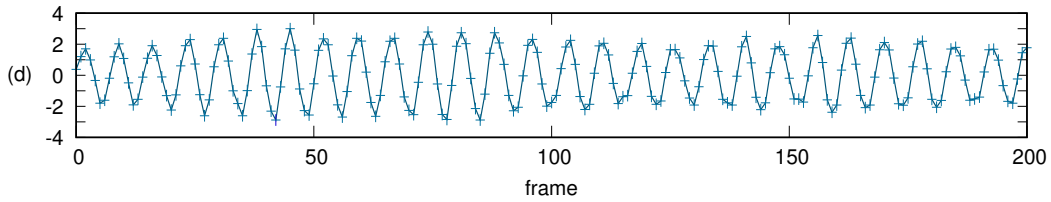
(a) 遅い動きにおける *of2bmflc* の出力



(b) 遅い動きにおける *bmflc* の出力



(c) 速い動きにおける *of2bmflc* の出力



(d) 速い動きにおける *bmflc* の出力

図 5.11: 手の動きから抽出した振動成分

(a) 入力画像　(b) 出力クラス画像　(c) 正解ラベル画像

図 5.12: 器具検出システムの処理結果の例

表 5.5: プロトタイプ手術器具検出用ネットワークの構成

層	名称	出力サイズ
-	Input	$3 \times 640 \times 480$
1	Initial Conv.	$8 \times 320 \times 240$
2-7	Bottleneck 3.1	$32 \times 160 \times 120$
	Bottleneck 3.2	$32 \times 160 \times 120$
8-16	Bottleneck 4.1	$64 \times 80 \times 60$
	Bottleneck 4.2	$64 \times 80 \times 60$
	Bottleneck 4.3	$64 \times 80 \times 60$
17-22	Bottleneck 5.1	$32 \times 160 \times 120$
	Bottleneck 5.2	$32 \times 160 \times 120$
23-28	Bottleneck 6.1	$8 \times 320 \times 240$
	Bottleneck 6.2	$8 \times 320 \times 240$
29	Last Conv.	$2 \times 640 \times 480$

うになり，振戦の検出精度が向上することが期待される．図 5.12 (b) が示すように，ソフトウェア実装された現在のプロトタイプシステムでは，手術器具のエリアを概ね正しく抽出できており，Pixel Accuracy においておよそ 96.63% の精度を達成している．これはつまり，評価データセット中の 96.63% の画素が正しく分類されていることを示している．

現在のところ，システムは表 5.5 に示す全 29 層の大規模な深層畳み込みニューラルネットワークを利用しており，パイプライン化された現在の FPGA を用いた振戦検出システムには，資源使用量の問題から実装することができない．しかしながら，オプティカルフローは *of2bmflc* モジュールにより平均化されて利用されるため，必ずしも高い検出精度は必要ないことから，ネットワークの小規模化，2 値化等の量子化技術の導入 [10] 等により，資源使用量を FPGA に完全パイプライン実装しうるレベルまで削減できる可能性はあると考えられる．

5.6　総括

本章では，手の生理的振戦を抑制するシステムの実現に向けた，動画像から振動成分をリアルタイムに検出するシステムを提案し，その設計および FPGA への実装を示した．Lucas-Kanade オプティカルフローと BMFLC によるバンドパスフィルタを，倍精度浮動小数点演算と遜色ない精度を達成できる小数部ビット幅を用いて，固定小数点演算によるパイプライン構造で実装している．その結果，VGA 解像度の動画像を 60 fps で処理できるスループットと，バンドパスフィルタ部分で 0.74 μsec の低レイテンシを実現できた．また，オプティカルフローのフレームごとの統計情報を用いて振動成分を推定することで，ノイズの影響を受けづらい，ロバストな振動検出を実現している．実際のカメラを用いた実証実験でこのアプローチの妥当性を検証したところ，周波数の低い振動成分を適切に除去しつつ，振戦を模擬した速い振動のみを検出できることが確認された．加えて，振動検

出精度を改善するために，畳み込みニューラルネットワークを用いた手術器具検出システムの導入を検討しており，GPU を用いたプロトタイプによる評価の結果，比較的高い精度で手術器具とその他の領域を分類できることが分かった．

今後の課題として，この手術器具検出システムを改良し，FPGA に統合できる水準までネットワーク規模を抑える必要がある．また，オプティカルフローの計算において不可欠なフレームバッファ部分に，何らかの画像圧縮技術を導入したり，超解像技術を応用したりすることで，資源使用量において支配的である Block RAM の使用量を抑制することも，より小さな FPGA の利用を可能とし，システムの利用範囲を広げられるという点で有益であると考えられる．また，今回行った定性的な実証実験だけでなく，定量的な指標に基づいた実験を追加で行い，さらに実際の機械的フィードバックシステムに組み込んで，振戦抑制効果を評価する必要がある．

第 6 章

結論

本稿では，低解像度画像から高解像度画像を再構成する超解像，腹腔鏡手術の支援を目的とした手術画像セグメンテーション，そして，振戦抑制の実現に向けた低レイテンシの画像ベース振動成分検出という 3 つのリアルタイム動画像処理システムについて，それぞれのアプリケーションが持つ特性に応じた計算構造を設計し，FPGA あるいは GPU を用いた実装および評価を行った．

超解像システムでは，水平および垂直反転を組み合わせることで低解像度空間から高解像度空間へのマッピングを行う反転手法を提案し，これに基づいて構築された畳み込みニューラルネットワークを，完全パイプライン構造で FPGA に実装した．その際，数値表現に Residue Number System (RNS) を導入することで，演算を LUT により効率的に実現し，資源使用量の削減を図った．また，活性化関数に Leaky ReLU を採用し，RNS に必要なダイナミックレンジの低減を実現した．PSNR および SSIM に基づく品質評価の結果，反転手法の導入により，事前拡大手法やサブピクセル再構成手法を用いる場合と比べ，同等のネットワーク規模でより高い品質が得られることが確認できた．またハードウェア実装では，Quarter HD (960×540) からフル HD (1920×1080) への超解像を 60 fps で実行でき，レイテンシも 300 μs と小さく抑えられていた．

手術画像セグメンテーションにおいては，Depthwise Separable Convolution を用いたエンコーダ・デコーダ構造のネットワークに，内包するサブネットワークを繰り返し利用する再帰的構造を導入し，さらに Stochastic Depth 正則化を組み合わせることで，小規模な学習データセットを用いた場合でも過学習の影響を抑制でき，分類精度の向上につながることが確認された．また，アップサンプリング畳み込み層に，超解像システムで提案した反転手法に基づくサブピクセル再構成を導入することで，一般的な転置畳み込みを用いる場合と比べ，より高い精度を実現できた．さらに，GeForce GTX 1080 を用いた性能評価では，腹腔鏡の自律制御に十分なフレームレートである 17 fps を達成した．

画像ベース振動成分検出では，Lucas-Kanade オプティカルフローと，位相遅れのないバンドパスフィルタとして機能する，Band-Limited Multiple Fourier Linear Combiner (BMFLC) を組み合わせた設計を行った．オプティカルフローのフレームごとの統計情報を用いることで，ノイズの影響を受けづらい頑健な検出を実現するとともに，深いパイプライン構造を持たせた FPGA 実装により，VGA 解像度 (640×480) で 60 fps のフレームレートと，BMFLC 部分で 0.74 μs という低レイテンシを達成した．実際にカメラを用いて実証実験を行ったところ，周波数の低い振動成分を適切に除去しつつ，振戦を模擬した速い振動のみを検出できることが確認されている．また，検出精度の改善のため，畳み

込みニューラルネットワークを用いた手術器具検出システムの導入も検討しており，GPUを用いたプロトタイプによる評価で，比較的高い精度での検出が可能であることを示した．

今後の展望として，超解像では各層におけるネットワーク構成パラメータの最適化，画像の大域情報を考慮するアーキテクチャの採用，Separable Convolution 導入による資源使用量の低減等が挙げられる．一方手術画像セグメンテーションでは，データセットの強化とネットワーク構造のさらなる改良，2 値化等の量子化等と組み合わせての FPGA 実装によるパフォーマンス改善と省電力化を目指す．また，ロボットと組み合わせた評価実験による安全性と有効性の検証も必須である．最後に画像ベース振動成分検出については，現時点では GPU での処理に留まっている手術器具検出を FPGA に統合するほか，フレームバッファの圧縮による Block RAM 使用量の削減，定量的な検出精度の評価等が求められる．

謝 辞

本研究の実施にあたって丁寧なご指導を賜り，また本稿の執筆につきましても様々な助言をしていただきました，長崎大学大学院工学研究科 情報工学コース 柴田裕一郎教授に，心より感謝いたします．また，友永航生様，藤田光暉様，上津原和也様，田原あかね様をはじめとした研究室の皆様方にも厚く御礼申し上げます．

参考文献

[1] I. J. Goodfellow, J. P. Abadie, M. Mirza, B. Xu, D. W. Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Proc. Neural Information Processing Systems (NeurIPS)*, Vol. 27, pp. 2672–2680, 2014.

[2] A. Radford, L. Metz, and S. Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. In *arXiv preprint arXiv:1511.06434*, 2016.

[3] F. Chollet. Xception: Deep Learning with Depthwise Separable Convolutions. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[4] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, and W. Wang, T. Weyand, M. Andreetto, and H. Adam. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*, 2017.

[5] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. Chen. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

[6] X. Zhang, X. Zhou, M. Lin, and J. Sun. ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 6848–6856, 2018.

[7] M. Courbariaux, Y. Bengio, and J. P. David. BinaryConnect: Training Deep Neural Networks with binary weights during propagations. In *Proc. Neural Information Processing Systems (NeurIPS)*, Vol. 2, pp. 3123–3131, 2015.

[8] I. Hubara, M. Courbariaux, D. Soudry, R. E. Yaniv, and Y. Bengio. Binarized Neural Networks. In *Proc. Neural Information Processing Systems (NeurIPS)*, pp. 4114–4122, 2016.

[9] D. Zhang, J. Yang, D. Ye, and G. Hua. LQ-Nets: Learned Quantization for Highly Accurate and Compact Deep Neural Networks. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 365–382, 2018.

[10] H. Nakahara, H. Yonekawa, T. Sasao, H. Iwamoto, and M. Motomura. A memory-based realization of a binarized deep convolutional neural network. In *Proc. International Conference on Field-Programmable Technology (FPT)*, pp. 277–280, 2016.

[11] C. Dong, C. C. Loy, K. He, and X. Tang. Learning a Deep Convolutional Network for Image Super-Resolution. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 184–199, 2014.

[12] C. Dong, C. C. Loy, K. He, and X. Tang. Image Super-Resolution Using Deep Convolutional Networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 38, No. 2, pp. 295–307, 2016.

[13] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang. Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 1874–1883, 2016.

[14] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 105–114, 2017.

[15] J. Su, D. V. Vargas, and K. Sakurai. One Pixel Attack for Fooling Deep Neural Networks. *IEEE Transactions on Evolutionary Computation*, Vol. 23, No. 5, pp. 828–841, 2019.

[16] 天野英晴（編）. FPGA の原理と構成. オーム社, 2016.

[17] A. Putnam, A. Caulfield, E. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, J. Gray, M. Haselman, S. Hauck, S. Heil, A. Hormati, J. Y. Kim, S. Lanka, E. Peterson, A. Smith, J. Thong, P. Y. Xiao, D. Burger, J. Larus, G. P. Gopal, and S. Pope. A Reconfigurable Fabric for Accelerating Large-Scale Datacenter Services. In *Proc. International Symposium on Computer Architecuture (ISCA)*, pp. 13–24, 2014.

[18] V. Nair and G. E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proc. International Conference on Machine Learning (ICML)*, pp. 807–814, 2010.

[19] X. Glorot, A. Bordes, and Y. Bengio. Deep Sparse Rectifier Neural Networks. In *Proc. Artificial Intelligence and Statistics (AISTATS)*, pp. 315–323, 2011.

[20] Y. LeCun, Y. Bengio, and G. Hinton. Deep Learning. *Nature*, Vol. 521, pp. 436–444, 2015.

[21] A. L. Maas, A. Y. Hannun, and A. Y. Ng. Rectifier Nonlinearities Improve Neural Network Acoustic Models. In *International Conference on Machine Learning (ICML) Workshop on Deep Learning for Audio, Speech, and Language Processing (WDLASL)*, 2013.

[22] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, Vol. 323, pp. 533–536, 1986.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2016.

[24] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *The Journal of Machine Learning Research*, Vol. 15, pp. 1929–1958, 2014.

[25] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *Proc. International Conference on Machine Learning (ICML)*, pp. 448–456, 2015.

[26] Y. LeCun and Y. Bengio. Convolutional networks for images, speech, and time series. In *The handbook of brain theory and neural networks*, pp. 255–258, 1995.

[27] A. Krizhevsky, I. Sutskever, and G. Hinton. Imagenet Classification with Deep Convolutional Neural Networks. In *Proc. Neural Information Processing Systems (NeurIPS)*, pp. 1106–1114, 2012.

[28] H. L. Garner. The Residue Number System. *IRE Transactions on Electronic Computers*, Vol. EC-8, No. 2, pp. 140–147, 1959.

[29] H. Nakahara and T. Sasao. A Deep Convolutional Neural Network Based on Nested Residue Number System. In *Proc. International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 1–6, 2015.

[30] International Telecommunication Union. BT.2020 : Parameter values for ultra-high definition television systems for production and international programme exchange. https://www.itu.int/rec/R-REC-BT.2020, accessed Mar. 2. 2018.

[31] NHK 放送技術研究所 研究内容紹介. http://www.nhk.or.jp/strl/vision/r1/r1.htm.

[32] H.265 : High efficiency video coding. https://www.itu.int/rec/T-REC-H.265.

[33] Cisco Systems, Inc. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016-2021 White Paper. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html, accessed Mar. 2. 2018.

[34] ディジタル画像処理 第 2 版, pp. 166–168. 公益財団法人 画像情報教育振興協会.

[35] W. T. Freeman, T. R. Jones, and E. C. Pasztor. Example-Based Super-Resolution. *IEEE Computer Graphics and Applications*, Vol. 22, No. 2, pp. 56–65, 2002.

[36] S. C. Park, M. K. Park, and M. G. Kang. Super-Resolution Image Reconstruction: A Technical Overview. *IEEE Signal Processing Magazine*, Vol. 20, No. 3, pp. 21–36, 2003.

[37] J. Kim, J. K. Lee, and K. M. Lee. Deeply-Recursive Convolutional Network for Image Super-Resolution. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, pp. 1637–1645, 2016.

[38] T. Manabe, Y. Shibata, and K. Oguri. FPGA Implementation of a Real-Time Super-Resolution System Using Flips and an RNS-Based CNN. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E101.A, No. 12, pp. 2280–2289, 2018.

[39] Preferred Networks, inc. Chainer: A flexible framework of neural networks. http://chainer.org/.

[40] OpenCV Team. OpenCV. https://opencv.org/.

[41] The scikit-image development team. scikit-image. http://scikit-image.org/.

[42] D. Kingma and J. Ba. Adam: A Method for Stochastic Optimization. In *Proc. International Conference on Learning Representations (ICLR)*, 2015.

[43] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, Vol. 13, No. 4, pp. 600–612, 2004.

[44] gnuplot homepage. http://www.gnuplot.info/.

[45] J. Hu, L. Shen, and G. Sun. Squeeze-and-Excitation Networks. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 7132–7141, 2018.

[46] J. Long, E. Shelhamer, and T. Darrell. Fully Convolutional Networks for Semantic Segmentation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

[47] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 39, pp. 2481–2495, 2017.

[48] O. Ronneberger, P. Fischer, and T. Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation. In *Proc. Medical Image Computing and Computer Assisted Intervention (MICCAI)*, pp. 234–241, 2015.

[49] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid Scene Parsing Network. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 6230–6239, 2017.

[50] F. Yu and V. Koltun. Multi-Scale Context Aggregation by Dilated Convolutions. In *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[51] G. Huang, Y. Sun, Z. Liu, D. Sedra, and K. Q. Weinberger. Deep Networks with Stochastic Depth. In *Proc. European Conference on Computer Vision (ECCV)*, pp. 646–661, 2016.

[52] S. Ioffe. Batch Renormalization: Towards Reducing Minibatch Dependence in Batch-Normalized Models. *Proc. Neural Information Processing Systems (NeurIPS)*, pp. 1945–1953, 2017.

[53] K. Turkowski. Filters for Common Resampling Tasks. In A. S. Glassner, editor, *Graphic Gems*, pp. 147–165. Academic Press, 1990.

[54] H. Zhang, K. Dana, J. Shi, Z. Zhang, X. Wang, A. Tyagi, and A. Agrawal. Context Encoding for Semantic Segmentation. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, pp. 7151–7160, 2018.

[55] B. D. Lucas and T. Kanade. An Iterative Image Registration Technique with an Application to Stereo Vision. In *Proc. International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 674–679, 1981.

[56] K. C. Veluvolu, U. X. Tan, W. T. Latt, C. Y. Shee, and W. T. Ang. Bandlimited Multiple Fourier Linear Combiner for Real-time Tremor Compensation. In *Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, pp. 2847–2850, 2007.

[57] C. N. Riviere, R. S. Rader, and N. V. Thakor. Adaptive Cancelling of Physiological Tremor for Improved Precision in Microsurgery. *IEEE Transactions on Biomedical Engineering*, Vol. 45, No. 7, pp. 839–846, 1998.

[58] E. Rocon, J. M. Belda-Lois, A. F. Ruiz, M. Manto, J. C. Moreno, and J. L. Pons. Design and Validation of a Rehabilitation Robotic Exoskeleton for Tremor Assessment and Suppression. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, Vol. 15, No. 3, pp. 367–378, 2007.

[59] A. As'arry, M. Z. Md zain, M. Mailah, and M. Hussein. Suppression of Hand Tremor Model Using Active Force Control with Particle Swarm Optimization and Differential Evolution. *International Journal of Innovative Computing, Information and Control (IJICIC)*, Vol. 9, No. 9, pp. 3759–3777, 2013.

[60] K. Sajith, V. Darade, and S. Chaudhuri. Hand Tremor Analysis Using Rigid Body Manipulation in a Dynamic Virtual Haptic Environment. In *Proc. Conference on Advances In Robotics (AIR)*, pp. 1–5, 2013.

[61] D. Case, B. Taheri, and E. Richer. Design and Characterization of a Small-Scale Magnetorheological Damper for Tremor Suppression. *IEEE/ASME Transactions on Mechatronics*, Vol. 18, No. 1, pp. 96–103, 2013.

[62] B. Soran, J. N. Hwang, S. I. Lee, and L. Shapiro. Tremor Detection Using Motion Filtering and SVM. In *Proc. International Conference on Pattern Recognition (ICPR)*, pp. 178–181, 2012.

[63] B. K. P. Horn and B. G. Schunck. Determining Optical Flow. *Artificial Intelligence*, Vol. 17, pp. 185–203, 1981.

[64] K. Cuppens, B. Vanrumste, B. Ceulemans, L. Lagae, and S. Van Huffel. Detection of Epileptic Seizures Using Video Data. In *Proc. International Conference on Intelligent Environments (IE)*, pp. 372–373, 2010.

[65] Z. Uhrikova, O. Sprdlik, V. Hlavac, and E. Ruzicka. Action Tremor Analysis from Ordinary Video Sequence. In *Proc. Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBS)*, pp. 6123–6126, 2009.

[66] B. Widrow and S. D. Stearns. *Adaptive Signal Processing*. Prentice Hall, 1985.

[67] R. J. Elble and W. C. Koller. *Tremor*. Johns Hopkins University Press, 1990.

[68] J. Porter, M. Thomson, and A. Wahab. Lucas-Kanade Optical Flow Accelerator. http://csg.csail.mit.edu/6.375/6_ 375_2011_www/handouts/other/group3_final.pdf, 2011.

[69] W. S. P. Fernando, L. Udawatta, and P. Pathirana. Identification of Moving Obstacles with Pyramidal Lucas Kanade Optical Flow and k means Clustering. In *Proc. International Conference on Information and Automation for Sustainability (ICIAFS)*, pp. 111–117, 2007.

[70] A. Tahara, Y. Hayashida, T. T. Thu, Y. Shibata, and K. Oguri. Power Performance Analysis of FPGA-Based Particle Filtering for Realtime Object Tracking. In *Proc. International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pp. 451–462, 2017.

[71] D. Bouget, R. Benenson, M. Omran, L. Riffaud, B. Schiele, and P. Jannin. Detecting Surgical Tools by Modelling Local Appearance and Global Shape. *IEEE Transactions on Medical Imaging* , Vol. 34, pp. 2603–2617, 2015.

[72] Detecing Surgical Tools by Modelling Local Appearance and Glocal Shape. https://dbouget.bitbucket.io/2015_tmi_surgical_tool_detection/.